SDC

TM-4717/000/00

THE APPLICATION OF GENERALIZED, CYCLIC, AND MODIFIED NUMERICAL INTEGRATION ALGORITHMS TO PROBLEMS OF SATELLITE ORBIT COMPUTATION

15 March 1971

SERIES BASE NO. / VOL. / REISSUE

TM-4717/000/00

TECHNICAL MEMORANDUM

(TM Series)

This document was produced by SDC in performance of contract NAS5-11785

THE APPLICATION OF GENERALIZED, CYCLIC, AND MODIFIED NUMERICAL INTEGRATION ALGORITHMS TO PROBLEMS OF SATELLITE ORBIT COMPUTATION

By Leonard Chesler and Sam Pierce

SYSTEM
DEVELOPMENT
CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA
90406

15 March 1971

Prepared By:

System Development Corporation
Santa Monica, California

For:

Goddard Space Flight Center
Greenbelt, Maryland

ABSTRACT

This document describes results of work performed on NASA Contract NAS5 11785 to develop and evaluate generalized, cyclic, and modified multistep numerical integration methods for application to problems of satellite orbit computation. Generalized methods have been compared with the presently utilized Cowell methods; new cyclic methods have been developed for special second-order differential equations; and several modified methods have been developed and applied to orbit computation problems. Special computer programs have been written to generate coefficients for these methods, and subroutines have been written which allow use of these methods with NASA's GEOSTAR computer program.

SUMMARY AND CONCLUSIONS

Numerical analysis plays a significant role in the modern theory of astrodynamics. Because of this numerical techniques have become an important area for research and development. This is particularly true in the area of satellite computation, where the equations describing satellite motion cannot, in general, be solved in analytic form. Mathematically these problems fall in the category of initial value problems and both multistep and single-step methods are applicable. However, the complexity of the force function usually requires that the number of force evaluations be kept to a minimum, thereby tending to favor multistep methods for practical problems.

Multistep methods have been used for many years. The best and most widely known methods are the Cowell and Gauss-Jackson type currently used at Goddard Space Flight Center. These familiar methods are limited by the Dahlquist [1] stability results which limit the order of accuracy that is achievable if the methods are to be stable.

Recently several new methods have been developed which bypass the Dahlquist restrictions. These include the generalized multistep methods of Butcher [2], Gregg and Stetter [3] and Dyer [6], the cyclic methods of Hansen [4], and the modified methods of Gear [5]. Because of their potential promise and because each of these methods offers a new and unique approach to multistep problems, it was felt to be essential that the feasibility of each method for application to orbit computation be thoroughly evaluated.

This study is part of this evaluation effort. This report describes the results of work performed under NASA Contract NAS5 11785. The report is divided into five sections. Section 2 presents the results of comparisons between off-grid generalized methods and the Cowell methods currently used at Goddard Space Flight Center. The test cases for these comparisons were nearly circular orbits at varying altitudes using both point mass and full

non-spherical potential force models. The starters employed with the generalized methods were both of the multistep and high order single-step (Runge-Kutta) class. Also, a new generalized method is developed which eliminates the off-grid point function evaluation. The theory for variable step procedures for generalized methods is also developed. Section 3 extends the cyclic theory of Hansen to special second-order equations and preliminary computational results are presented. In Section 4, several modified methods are derived and preliminary computational results are discussed. In Section 5, a general analysis is developed which can be used to improve all of the methods studied, including the presently utilized Cowell method.

Computer programs have been developed for generating coefficients of the developed generalized, cyclic, and modified multistep methods. Descriptions of these programs are presented in Appendices A-C. Computer programs and subroutines for implementing generalized, cyclic, and modified methods on NASA's GEOSTAR program have also been written. Descriptions of these subroutines are presented in Appendix D.

Results of comparisons between 11th order Cowell and off-grid generalized methods using a multistep starter indicate that the generalized methods tend to do better than the Cowell methods for high altitude satellites (e.g., ATS, semi-major axis 42,166,307.5 meters, eccentricity .0003), while the Cowell methods are superior at low altitudes. Further, if a Runge-Kutta starter is employed with the generalized method, the accuracy advantage of the generalized scheme increases considerably for large stepsizes, resulting in a marked superiority of this scheme over the Cowell method for both medium- and high-altitude satellites, while remaining competitive at low altitudes. It should be noted, however, that an erratic error-stepsize behavior has been observed, and that these results have been obtained only for the class of nearly circular orbit so that further studies are still required. Also, the generalized methods still have to be compared

to high order Cowell methods (12th order and greater) before final conclusions can be drawn. At low altitudes additional comparisons should be made with low-order generalized methods. Preliminary results are good for the new generalized method in which the off-grid derivatives are interpolated. Further evaluations are required. If successful, these methods will also be competitive for medium and high altitude satellites. These recommendations, however, should not deter from the fact that the "best" method presently available to the authors for integrating the circular class of orbits considered in this report is the off-grid generalized method at an appropriate stepsize coupled with a high order single-step starter.

Preliminary results for the cyclic methods developed in Section 3 are extremely good. Indications are that the cyclic approach holds considerable promise for orbit computation applications. Results for the modified method are disappointing. However, several changes are possible which could improve the method's performance.

In summary, as a result of our present study, the following recommendations are made:

- Thoroughly evaluate the off-grid generalized methods coupled with the Runge-Kutta type starters for medium and high eccentricity orbits using the procedures developed in Paragraph 2.5.
- Develop an off-grid multistep starter.
- Analyze the behavior of the off-grid Runge Kutta scheme with respect to error vs. stepsize.
- Thoroughly evaluate the new generalized method which eliminates the off-grid force evaluation. Compare this method with 11th and greater order Cowell methods for

medium and high altitude satellites using the GEOSTAR full force model.

- Perform further comparisons with the Cowell method of lower order generalized methods (k=5, and k=4) for low and medium altitude satellites.
- Develop high order cyclic methods and compare with the presently utilized Cowell methods.
- Perform the analysis and implement the improvements for modified methods suggested in Paragraph 4.7.
- Perform an analysis similar to that described in Section 5 for the Cowell, cyclic, and generalized methods.

1. INTRODUCTION

In the area of orbit dynamics the use of numerical techniques is virtually mandatory. In practical problems equations yielding to analytic solution are the exception rather than the rule. Numerical analysis techniques have therefore come to play an important role in the modern solution of orbit dynamic problems and as such have become, in themselves, a significant area for further research and development.

This is particularly true of problems involving numerical integration and the solution of differential equations. This is due to the central role played by the orbit determination and computation in a large number of satellite flight support computer programs. Most of these programs depend in part on achieving accurate and efficient solutions to the equations describing the motion of a satellite either around the earth or some other planetary body. Unfortunately, except for the simple two-body case, these equations cannot be readily solved analytically. For most practical applications, the problem must be solved using numerical techniques.

The motion of a satellite can be described in terms of a set of first- and second-order differential equations subject to a specified set of initial conditions. In general, the equations of motion are highly complex, involving non-spherical gravitation effects, possibly drag and solar radiation, and the effects of other bodies, such as the sun and moon. Mathematically the problem falls in the category of "initial value" problems and both multistep and single-step methods are applicable. However, the complexity of the force model usually makes it imperative that the number of force evaluations be kept to a minimum to achieve economical computation times. This requirement tends to dictate the use of multistep integration methods as opposed to single-step methods of the Runge-Kutta type.

Multistep methods have been used for many years for solving orbit computation problems and several classical methods exist. The best and most widely used

Of these methods are the Cowell and Gauss-Jackson algorithms. Currently, NASA at Goddard Space Flight Center uses a Cowell-type method which is essentially equivalent to the Gauss-Jackson approach.

In the multistep approach to solving differential equations, integration is performed using a polynomial fit to past values of the solution and its derivatives. Thus, an approximation to the solution of $\ddot{y} = f(t,y(t),\dot{y}(t)$ at the point $t+h$ is obtained using knowledge of the solution and the derivatives at the past values $t,t-2h,\ldots,t-kh$. In this case h is the step size and $k+1$ back points are utilized.

The basic measures of any numerical technique are its accuracy of approximation and the computation time. In multistep methods computation time is generally directly proportional to the stepsize used since this determines the number of computations performed by the method in a specified interval of integration. Accuracy is related to the degree of the polynomial approximation, which is a function of the number of back points used, and to the stability of the method. The degree of the polynomial determines the quality of the approximation to the underlying differential equation. Stability refers to properties of the approximating algorithm which can cause systematic errors to grow without bound when an unstable method is used. Stable methods have the property that errors are damped out as the computation proceeds. In summary, it is desirable to develop methods which allow the use of large step size, use a high-degree polynomial, and are stable.

The Cowell methods presently used by NASA are based on Newtonian- type formulas. These traditional multistep methods are limited by the familiar Dahlquist stability results [1]. This severely limits the order of accuracy (i.e., the degree of polynomial for which the approximation is exact) achievable if the methods are to be convergent. Recently several new types of multistep methods which bypass the Dahlquist restrictions have been developed. These include the generalized multistep methods of Butcher [2], Gregg and Stetter [3], and Dyer [6], the cyclic method of Hansen [4], and the modified methods of Gear [5].

Each of these new approaches produces strongly stable methods which have greater accuracy than the classical methods for a given number of back points.

Because of their potential promise and because these methods offer several new approaches to the problem, it was felt to be essential that their feasibility for application to orbit computation be thoroughly evaluated. This study is part of this evaluation effort. This report describes the results of work performed under NASA Contract NAS5 11785. Each of these three methods has been specialized and extended for application to satellite orbit computation problems. Algorithms and computer programs for deriving the coefficients of the methods have been developed. Subroutines for implementing the newly developed methods have been implemented in NASA's GEOSTAR computer program and where feasible comparisons have been made with existing Cowell methods.

Section 2 discusses 11th order off-grid generalized multistep methods and the results of comparisons with the present 11th order Cowell methods in use. Using a multistep starter, the generalized methods show promise for high-altitude satellites while the Cowell methods are superior at low altitudes. However, with a Runge-Kutta starter, the generalized methods are superior for medium and high altitude satellites while remaining competitive at low altitudes. It should be noted, however, that these results have been obtained with respect to a circular class of orbits only and additional comparisons should be made using medium and high eccentricity orbits. Also, a "periodic" behavior of the error as the stepsize is varied, has been detected which should be analyzed. Further, the generalized methods should be compared with higher-order Cowell methods (12th or greater) before final conclusions can be drawn, but this should not deter from the fact that, at the current time, the most effective technique to integrate circular, medium and high altitude orbits known to the authors is the off-grid generalized method with an appropriately large stepsize using a Runge-Kutta starter.

A new generalized method has been developed which uses an interpolated value for the off-grid acceleration point. Initial results are promising. This method should be thoroughly evaluated. If successful it will significantly improve the generalized method's performance at medium- and high-altitudes using the multistep starter.

In Section 3, the theory developed for cyclic methods [4] is extended to special second-order differential equations. The resulting methods show considerable promise. It is recommended that higher order cyclic methods be developed and compared with the present Cowell methods.

The modified methods are discussed in Section 4. Results to date are disappointing and further analysis is required to determine if the methods can be improved. Present indications are that considerable improvement is possible. Requirements for this analysis are discussed in Paragraph 4.7.

Appendices A-D describe the computer programs developed to support the analysis. Listings and source decks for each of the described programs have been sent to NASA under separate cover.

The presently used Cowell methods apparently owe their success in part to the coupled action of Class II methods, for predicting and correcting position, and Class I methods, for predicting and correcting velocity. Class II methods are methods for solving the special second-order equation $\ddot{y}=f(t,y)$ and Class I methods are methods for solving the first-order equation $\dot{y}=g(t,y)$. We have followed this approach in developing computation algorithms for all of the methods discussed.

2. GENERALIZED MULTISTEP METHODS

2.1 INTRODUCTION

Generalized multistep or off-grid methods were first described for first-order equations by Gregg and Stetter [3] and Butcher [2]. These methods have been extended by Dyer [6] to Class II equations, i.e., special second-order equations of the form $y'' = f(x,y)$. Dyer's paper extends the Hermite interpolation approach presented by Butcher and develops difference equations based on what are termed quasi-Hermite interpolating polynomials. In some ways the theory parallels standard Hermite techniques but there are significant differences.

Using this approach it is possible to develop Class I and II algorithms for performing orbit computations. The off-grid methods are of significant interest because, along with the modified and cyclic methods discussed later in this report, they have the distinction of bypassing the classical Dahlquist stability results.[1] This allows the use of higher-order polynomials with fewer back points than the classical methods. This property coupled with favorable preliminary results [6] made it important that the off-grid method be evaluated for potential orbit computation applications.

To date several off-grid algorithms have been developed and implemented on NASA's GEOSTAR orbit computation program. Extensive comparisons have been made between these and the presently used Cowell methods.

The results indicate that an 11th order off-grid method coupled with an 8th order single-step starter is superior to an 11th order Cowell process for the medium and high altitude orbits tested and performs equally well with the low altitude orbit. Further, a scheme consisting of the 11th order off-grid with a 12th order multistep starter also tended to do better than Cowell for higher altitude satellites although losing its superiority for low altitude satellites. Another phenomenon which was observed is that in general the 11th order off-grid method behaved similar to a higher order Cowell mthod in the sense that its advantage was greatest in a simple force model or high altitude environment and least in a complex force model or low altitude environment. This phenomenon which occurs in both off-grid and high-order Cowell methods requires further study involving consideration of the stability and error properties of the method. It is recommended that such a study be made following the general approach presented in Section 4 of this report.

Further detailed discussion of the comparisons between the off-grid and Cowell methods is presented in Paragraph 2.3.

A basic problem with the off-grid methods is that they require an extra function evaluation at the off-grid point resulting in three evaluations per step. This considerably degrades the methods computation effectiveness when compared to the Cowell methods since these can be designed to require one to two evaluations per step. An approach has been developed to eliminate much of the computation required for the off-grid function evaluation, and was implemented on the GEOSTAR program. Results to date with this algorithm are encouraging, but more work is needed on these methods to improve accuracy. Preliminary computations indicate that these methods will be competitive with the Cowell and standard off-grid methods for the medium- to high-altitude satellites (GEOS and ATS).

Paragraph 2.4 presents a discussion of the new method and suggestions for its possible improvement.

In addition to the low-eccentricity orbits (Delta PAC, GEOS, ATS) presently evaluated, a variable-step procedure has been developed which will allow an evaluation of off-grid techniques for use with high-eccentricity orbits. The developed procedures are based on a generalization of the local error techniques in [7] and a generalization of Milne's approximation for high-order derivatives in [8]. These procedures have been implemented on the GEOSTAR program and after being properly calibrated can be used to evaluate off-grid methods for high-eccentricity orbits. Variable-step procedures are discussed in detail in paragraph 2.5.

2.2 INTEGRATION FORMULAS

2.2.1 Generalized Methods

Multistep methods are used to approximate solutions to Class I or II equations of the forms:

Class I	Class II	
$\dot{y} = f(t,y)$	$\ddot{y} = g(t,y)$	(2-1)

assuming the solution $y(t_{n-i})$ is known for $i = 1, --, k$ equally spaced back points. Traditional Class II k-step methods are defined by difference equations of the form

$$\sum_{i=0}^{k} a_i y(t_{n-i}) + h^2 \sum_{i=0}^{k} b_i \ddot{y}(t_{n-i}) = 0 \qquad (2\text{-}2)$$

where h is the step size (interval). For Class I, h^2 is replaced by h and the second derivative is replaced by the first derivative; for a predictor, $b_o = 0$,

The generalized or off-grid methods use an extra derivative evaluation at an off-grid point. The Class II corrector then takes the form

$$\sum_{i=0}^{k} a_i y(t_{n-i}) + h^2 \sum_{i=0}^{k} b_i \ddot{y}(t_{n-i}) + h^2 b_\theta \ddot{y}(t_{n-\theta}) = 0 \qquad (2\text{-}3)$$

where $0 < \theta < 1$. For Class I, the second derivatives are replaced by first derivatives and h replaces h^2.

The off-grid methods are based on the use of quasi-Hermite interpolation polynomials to construct the difference equations and derive stable methods. The coefficients of a method are expressed in terms of quasi-Hermite interpolating functions and a value of θ is chosen which ensures the stability of the method.

The term quasi-Hermite is used here to designate polynomials which are determined by imposing derivative conditions, which are not necessarily consecutive, at points of the interpolating set. This kind of interpolation results naturally when one wishes to replace the Class II equation $\ddot{y} = g(t,y)$ by a difference equation relating functional values and second derivative values but does not restrict the first derivative. The same procedures can also be used to derive new Class I methods (i.e., for $\dot{y} = f(x,y)$). Procedures are for Class I methods are illustrated below. Class II procedures are basically the same. A description of the computer programs developed to generate coefficients for Class I and II methods is presented in Appendix A.

To see how the interpolating polynomials and, thus, difference equation coefficients are derived, assume an interpolating set o, h, 2h, --, kh and write the determinantal form

$$
\begin{array}{lccccc}
P(hx) & 1 & hx & h^2x^2 & - - - - & h^{2k+1}x^{2k+1} \\
f(0) & 1 & 0 & 0 & - - - - & 0 \\
f(h) & 1 & h & h^2 & - - - - & h^{2k+1} \\
 & & & & & \\
f(kh) & 1 & hk & h^2k^2 & - - - - & h^{2k+1}k^{2k+1} \\
f^1(0) & & 1 & 0 & - - - - & 0 \\
f^1(h) & 0 & 1 & 2h & - - - - & (2k+1)h^{2k} \\
\begin{matrix}1\\1\end{matrix} & & & & & \begin{matrix}1\\1\end{matrix} \\
f^1(kh) & & 1 & 2kh & - - - - & (2k+1)h^{2k}k^{2k}
\end{array}
$$

in which differentiation is with respect to $\bar{x} = hx$. Every row except the first represents a functional or derivative condition placed on P. If these conditions determine P uniquely (as the conditions shown always do),

and if P is identified with f, it is easy to see that

$$\begin{array}{lccccc} P(hx) & 1 & x & x^2 & \cdots & x^{2k+1} \\ P(0) & 1 & 0 & 0 & \cdots & 0 \\ P(h) & 1 & 1 & 1 & \cdots & 1 \end{array}$$

$$\begin{array}{lccccc} P(kh) & 1 & k & k^2 & \cdots & k^{2k+1} \\ hP^1(0) & 0 & 1 & 0 & \cdots & 0 \\ P^1(h) & 0 & 1 & 2 & \cdots & 2^{k+1} \\ & & & & & 1 \\ hP^{1\,1}(kh) & 0 & 1 & 2k & & (2k+1)k^{2k} \end{array} \equiv 0$$

Expanding by the first column

$$P(hx) + \sum_{i=0}^{k} H_i(x)\, f(ih) + h \sum_{i=0}^{k} \bar{H}_i(x) f^1(ih) \equiv 0$$

this equation can be used as a predictor and by differentiating with respect to x, it can be used as an off-grid corrector with $x = k-\theta$ (e.g., compare the Class I method corresponding to Eq 2-3). Rows may be removed from determinantal forms for conditions not imposed. Corresponding terms are removed from the difference equation. It is necessary that conditions imposed uniquely determine the polynomial in question.

In practice, in the differentiated form, θ was varied until stability was achieved, i.e., until the extraneous roots of $\sum_{i=0}^{k} H_i(x) r^{k-i} = 0$ were within the unit circle. Class II equations are derived similarly. We have chosen the first interpolating point at the origin. This choice is arbitrary since the weighting functions H_i, $\bar{H}_i$ are independent of translation as well as stepsize.

A detailed description of the specific methods used in our work is provided in Appendix A.

Off-grid methods for k=4,5, and 6 have been developed. Coefficients for these methods are presented in Section 2.6. The order of the methods are 8, 10, and 11, i.e., the methods are exact for polynomials of degree 9, 11 and 12. The k=6 method is discussed in the next paragraph.

The approach taken in developing off-grid algorithms parallels that for the present Cowell techniques in the sense that the equation

$$\ddot{y} = f(t,y,\dot{y})$$

is integrated by a Class II method to obtain positions while velocity is calculated using a Class I method.

The formulas required by the k=6 off-grid algorithm are

Class II Predictor
$$y_{n+1} = \sum_{i=0}^{k} a_i y_{n-i} + h^2 \sum_{i=0}^{k} b_i^{(1)} \ddot{y}_{n-i}$$

Class I Predictor
$$\dot{y}_{n+1-\theta} = \sum_{i=0}^{k} a_i^{(2)} \dot{y}_{n-i} + h \sum_{i=0}^{k} b_i^{(2)} \ddot{y}_{n-i}$$

Class II Predictor
$$y_{n+1-\theta} = \sum_{i=0}^{k} a_i^{(3)} y_{n-i} + h^2 \sum_{i=0}^{k} b_i^{(3)} \ddot{y}_{n-i}$$

Evaluate $\ddot{y}_{n+1-\theta} = f(t_{n+1-\theta}, y_{n+1-\theta}, \dot{y}_{n+1-\theta})$

Class I Corrector
$$\dot{y}_{n+1} = \sum_{i=1}^{k} a_i^{(4)} \dot{y}_{n+1-i} + h \sum_{i+0}^{k} b_i^{(4)} \ddot{y}_{n+1-} + h\, b_\theta^{(4)} \ddot{y}_{n+1-\theta}$$

Evaluate $\ddot{y}_{n+1} = f(t_{n+1}, y_{n+1}, \dot{y}_{n+1})$

Class II Corrector
$$y_{n+1} = \sum_{i=0}^{k} a_i^{(5)} y_{n+1-i} + h^2 \sum_{i=0}^{k} b_i^{(5)} \ddot{y}_{n+1-i} + h^2 b_\theta^{(5)} \ddot{y}_{n+1-\theta}$$

Evaluate $\ddot{y}_{n+1} = f(t_{n+1}, y_{n+1})$

hence requiring 3 function evaluations per step.

The required coefficients are given b_y

Six-Step Algorithm (11th Order $\theta = .21$

Class II coefficients in the correction of y_{n+1}

(1)		(1)	
a_1 =	1.676387251401263	b_0 = -	0.06709122286499903
a_2 =	0.2784005958059899	b_θ =	0.2013290453349816
a_3 = -	1.503376712957843	b_1 =	0.7840079304568607
a_4 =	0.4660026328926644	b_2 =	0.324155404103746
a_5 =	0.08258623285792565	b_3 = -	0.4947757672957449
		b_4 = -	0.1340343393756
		b_5 = -	0.003739633226947656

Class I coefficients in the prediction of $y_{n+1-\theta}$

(2)		(2)	
a_0 = -	61.05414468008109	b_0 =	5.008599855928512
a_1 = -	108.6964799519827	b_1 =	77.72951140163380
a_2 =	437.4446492674644	b_2 =	187.2900991236687
a_3 = -	291.8057707897887	b_3 =	0.0
a_4 =	15.40667458392008	b_4 = -	12.64660973709083
a_5 =	11.63263397206795	b_5 =	1.628245578848980
a_6 = -	1.927562401600067	b_6 =	0.1098656787133217

Class I coefficients in the prediction of $\dot{y}_{n+1-\theta}$

a_0 =	20.4653924088754	b_0 =	0.0
a_1 =	652.2720594267644	b_1 = -	243.8697318890013
a_2 =	1053.823369950025	b_2 =	-1303.843548629965
a_3 =	-1190.842745677504	b_3 =	-1279.762005515427

$a_4 = -759.4412318742384$ $b_4 = 0.0$

$a_5 = 193.4311707788693$ $b_5 = 125.6553886244422$

$a_6 = 31.29198498720788$ $b_6 = 7.143296024894652$

Class II coefficients in the prediction of $y_{n+1-\theta}$

$a_0^{(3)} = -19.08331076173265$ $b_0^{(3)} = 1.999528738448396$

$a_1 = -33.40992721597432$ $b_1 = 25.45651832752276$

$a_2 = 138.1645825076111$ $b_2 = 59.65810788083460$

$a_3 = -91.24955144201997$ $b_3 = 0.0$

$a_4 = 2.409417295666085$ $b_4 = -5.028433251450995$

$a_5 = 4.610990354550493$ $b_5 = 0.2448617876908021$

$a_6 = -0.4422007381008424$ $b_6 = 0.02714208741422536$

Class I coefficients in the correction of $\dot{y}_{n+1}$

$a_1^{(4)} = 0.4694598916997084$ $b_0^{(4)} = 0.0$

$a_2 = 1.742172782222655$ $b_\theta = 0.5296256656290971$

$a_3 = -0.630857158201849$ $b_1 = 0.8452828232809184$

$a_4 = -0.8481422202815336$ $b_2 = -0.3527199004940484$

$a_5 = 0.2272510305760354$ $b_3 = -1.140183151083101$

$a_6 = 0.04011567398498343$ $b_4 = 0.0$

$b_5 = 0.1543675679842099$

$b_6 = 0.009241291886342169$

Class II coefficients in the correction of y_{n+1}

(5)

$a_1 = 1.756753635776075$

$a_2 = -0.4518995288497442$

$a_3 = -0.8409215653309587$

$a_4 = 0.9519705800547516$

$a_5 = -0.3573465275980059$

$a_6 = -0.05855659405211152$

(5)

$b_0 = -0.03208160400375142$

$b_\theta = 0.17802152694308850$

$b_1 = 0.7533104176396140$

$b_2 = 0.3311535033971168$

$b_3 = 0.0$

$b_4 = 0.3845680319594847$

$b_5 = 0.09708107156156632$

$b_6 = 0.002601983778739063$

2.2.2 Cowell Methods

The Class II and I Cowell methods used for this study are of the form

(Class II) $$y_{n+s} = h^2 \left[{}^{II}S_n + (s-1)\,{}^{I}S_n + \sum_{i=o}^{k} b_i(s) y_{n-i} \right]$$

(Class I) $$\dot{y}_{n+s} = h \left[{}^{I}S_n + \sum_{i=o}^{k} C_i(s) y_{n-i} \right]$$

where ${}^{I}S_n$, ${}^{II}S_n$ are the first and second "sums," defined by the relations

$$\nabla\, {}^{I}S_n = {}^{I}S_n - {}^{I}S_{n-1} = \ddot{y} \qquad (2\text{-}5)$$

$$\nabla\, {}^{II}S_n = {}^{I}S_n, \qquad (2\text{-}6)$$

and s is a parameter to be specified. If s = 0, these equations are correctors, and s = 1 yields predictors. The coefficients $b_i(s)$, $c_i(s)$ can be derived by standard finite difference operator techniques.*

The formulas required by the algorithm are

(Class II Predictor) $$y_{n+1} = h^2 \left[{}^{II}S_n + \sum_{i=0}^{k} b_i^{(1)} y_{n-i} \right]$$

(Class I Predictor) $$y_{n+1} = h \left[{}^{I}S_n + \sum_{i=o}^{k} b_i^{(2)}\, y_{n-i} \right]$$

Evaluate $\ddot{y}_{n+1} = f(t_{n+1}, y_{n+1}, \dot{y}_{n+1})$

*Velez, C.E., Maury, J. L., "Derivation of Newtonian-Type Integration Coefficients and Some Applications To Orbit Calculations," Goddard Space Flight Center, Greenbelt, Maryland 20771, NASA TN D5958, October 1970.

(Class II Corrector) $$y_{n+1} h^2 \left[{}^{II}S_n + \sum_{i=0}^{k} b_i^{(3)} \ddot{y}_{n+1-i} \right]$$

(Class I Corrector) $$\dot{y}_{n+1} = h \left[S_n + \sum_{i=0}^{k} b_i^{(4)} \ddot{y}_{n+1-i} \right]$$

After applying these formulas the next step is to test and respect corrections if required, beginning with re-evaluation of y_{n+1}. We see that a step could be performed with only 1 function evaluation per step. The required coefficients are given by Maury and Brodsky.

*Maury, J. L., Brodsky, G. P., "Cowell Type Numerical Integration as Applied to Satellite Orbit Computation," Goddard Space Flight Center, Greenbelt, Maryland, NASA X553-69-46, December, 1969.

2.2.3 Starting Procedures

Two starting procedures were utilized in this study for the off-grid methods. The first is a 12th order iterative scheme based on the general Class II and Class I formulas given in Section 2.2.2. If we denote the required starting values by

y_i, $\dot{y}_i$, $i = \pm 1, \pm 2, -- \pm 5$, where

y_o, $\dot{y}_0$ are the given initial conditions, and if

$y_i^{(e)}$, $\dot{y}_i^{(e)}$

denote the 1th approximation of these values, the $(1+1)^{th}$ approximation is computed as

$$y_i^{(1+1)} = h^2 \left[II_{S_5} + (i=6)\ {}^{I}S_5 + \sum_{k+0}^{10} b_k (i-5) \ddot{y}_{5-k}^{(1)} \right]$$

$$\dot{y}_i^{(1+1)} = h \left[I_{S_5} + \sum_{k+0}^{10} C_k (i-5) \ddot{y}_{5-k}^{(1)} \right]$$

$$\ddot{y}_i^{(1+1)} = f(t_i, y_i^{(1+1)}, y^{(1+1)}) \qquad i = \pm 1 \ \pm 2, -- \pm 5$$

and repeated until convergence. The first approximation (1 = 1) is computed using 2-body analysis, details are given in the GEOSTAR II document*

*"GEOSTAR II, A GEOPOTENTIAL AND STATION POSITION RECOVERY SYSTEM," by Velez, Brodsky, NASA X-55370372, October 1970.

The second starter used with the off-grid methods is a 8th order, 10 evaluation Class I Runge-Kutta formula of the form

$$y_{n+1} = y_n + h \sum_{i=0}^{9} c_i k_i$$

$$k_i = f(t_n + a_i h,\ y_n + \sum_{J+0}^{i-1} b_{ij} k_j)$$

$$i = 1,2, \text{ -- } 9$$

where a_i, b_{ij}, c_i are constants defining the method and are derived by Shanks.* The stepsize selected for the starter was taken to be 1/2 the multistep stepsize.

For the Cowell methods, only the multistep starter was used, since it was determined that the Runge-Kutta starter did not improve the performance of the algorithm.

*Shanks, E., "Solution of Differential Equations by Evaluation of Functions," Math Composition, Volume 20, pp. 21-38, 1966.

2.3 DISCUSSION OF RESULTS

Extensive evaluations have been performed comparing off-grid and Cowell methods. Computations were performed using the GEOSTAR stand-alone orbit-generator program implemented at Goddard Space Flight Center on the IBM 360/67 computer.

Results of comparisons between 11th order Cowell and off-grid generalized methods, using a multistep starter, indicate that the generalized methods tend to do better than the Cowell methods for high eccentricity .0003), while the Cowell methods are superior at low altitudes. Further, if a Runge-Kutta starter is employed with the generalized method, the accuracy advantage of the generalized scheme increases considerably for large stepsizes, resulting in a marked superiority of this scheme over the Cowell method for both medium- and high-altitude satellites, while remaining competitive at low altitudes. It should be noted, however, that an erratic error-stepsize behavior has been observed, and that these results have been obtained only for the class of nearly circular orbits so that further studies are still required. Also, the generalized methods still have to be compared to high order Cowell methods (12th order and greater) before final conclusions can be drawn. At low altitudes additional comparisons should be made with low-order generalized methods. Preliminary results are good for the new generalized method in which the off-grid derivatives are interpolated. Further evaluations are required. If successful, these methods will also be competitive for medium and high altitude satellites.

Computational results for three representative satellites are illustrated in Figures 1-18. The three are the DELTA-PAC, GEOS B, and ATS satellites. A description of each is presented in Table 1. DELTA-PAC is low altitude; GEOS B, medium altitude; and ATS, high altitude. All are circular in shape with low values of eccentricity allowing the use of fixed-step methods.

Table 1. Satellite and Force Model Descriptions

Name	Semi-Major Axis (meters)	Eccentricity
DELTA-PAC	6,932,610.0	.00023000
GEOS B	8,070,920.2	.071910451
ATS	42,166,307.5	.000378974

Elliptic Motion Force Model

$$\ddot{\bar{r}} = -\frac{u\bar{r}}{r^3}$$

where

$\bar{r}$ = geocentric satellite position vector

r = magnitude of r

u = earth's gravitation constant

Full Force Model

$$\ddot{\bar{r}} = -\frac{u\bar{r}}{r^3} + \ddot{\bar{r}}_{NS} + \ddot{\bar{r}}_{PM} + \ddot{\bar{r}}_{SR} + \ddot{\bar{r}}_{D}$$

where

$\ddot{\bar{r}}_{NS}$ = acceleration due to the non-spherical potential of the earth given by

$$\ddot{\bar{U}} = \frac{u}{r} \sum_{n=0}^{15} \sum_{m=0}^{n} \left(\frac{ae}{r}\right) \left[C_{nm} \cos m\lambda + S_{nm} \sin m\lambda\right] R_n^m (\sin \psi)$$

a_e = semi major axis of the earth

λ = geocentric longitude of satellite

ψ = geocentric latitude of satellite

$P_n^m \equiv$ associated Legendre polynomials

$\ddot{\bar{r}}_{PM}$ = point mass gravitational effects of the sun and moon

$\ddot{\bar{r}}_{SR}$ = acceleration effects due to solar radiation

$\ddot{\bar{r}}_{D}$ = acceleration effects due to drag.

A variable-step procedure for off-grid methods has been developed and is described in Paragraph 2.4. If successful, it will allow evaluation of off-grid techniques for high-eccentricity satellites. This latter analysis would be an important extension to the present work since off-grid methods might be highly effective on the smoother portions of such orbits.

Figures 1 - 6 present results for the GEOS satellite. Figures 1 and 2 show the relation between the error of integration and stepsize for the off-grid k = 6 (11^{th} order) and Cowell 11^{th} order methods. Errors are presented for 30-day arcs. Figure 1 illustrates the significant degradation of results that takes place between the elliptic (two-body force model) case and the full-force case when using the multistep starter. The Cowell method performs equally (slightly better for full force) in both cases.

Figure 1 illustrates the mocked improvement in off-grid results possible when using the Runge-Kutta starter. Results for the Cowell method did not appreciably change with this starter. For steps sizes larger than 250 seconds both methods tended to be unstable.

Figure 3 illustrates the superiority of the off-grid method at larger stepsizes. A nominal error of 10 meters/week of arc is used to define acceptable operating regions. Using this criterion, the maximum allowable stepsize for the Cowell method is about 110 secs. For the off-grid method using the Runge-Kutta starter, the step size for both elliptic and full force (cases) is greater.

Figure 4 plots the total number of force evaluations required to achieve a specified level of accuracy. For the error criterion of 10 meters/week of arc the off-grid method, using the Runge-Kutta starter, is superiod for both the full force and the elliptic case.

Figure 4 illustrates the effect that a significant reduction in the number of force evaluations could have. Such a reduction might be possible when the off-grid forces are interpolated rather than evaluated (preliminary results indicate that this is possible; see Paragraph 2.4). This technique, if successful, would further improve the off-grid method for this type of satellite. Figures 5 and 6 illustrate typical propagated error curves for the GEOS B satellite.

Figures 7 - 12 present results for the DELTA PAC satellite using the multistep starter. Significant degradation takes place between the elliptic and full-force cases. However, this is also true for the Cowell method though to a lesser degree. For the elliptic case, the off-grid method

allows considerably higher stepsizes than the Cowell method while in the full force case, stepsizes are practically the same for both methods. For the nominal error criterion off-grid step sizes of greater than 225 secs. are allowable for the elliptic case while the maximum for full forces is 67 secs. The maximum stepsizes for the Cowell method are 97 secs. for elliptic motion and 65 secs. for the full-force case. In terms of the required number of force evaluations for a specified level of accuracy the Cowell method is superior in both the elliptic and full forces case. Later results using the Runge-Kutta starter somewhat change this picture indication that off-grid will be competitive with Cowell even for this low altitude satellite. Figures 13 - 18 present results for the ATS satellite using a 60-day arc and the multistep starter. Results with the Runge-Kutta starter were substantially the same. Figure 15 illustrates that far greater stepsizes are possible with the off-grid method than with the Cowell method for both the elliptic and full force cases. Figure 16 indicates that this advantage still holds even when the number of force evaluations is considered. For the ATS satellite within the ranges of accuracy considered the off-grid k = 6 (11^{th} order) method is superior to the Cowell 11th order method.

The next question is whether Cowell methods of higher order (12, 13, etc.) might also be superior for the ATS type satellite. A preliminary analysis using a 12^{th}-order Cowell method seems to indicate this is true. In fact the 12^{th}-order Cowell seems to exaggerate the strengths and weakensses of the off-grid method. Thus the 12^{th}-order Cowell performs very well for the elliptic versions of PAC and GEOS B but is considerably degraded for both full force cases. The method performs very well in both categories for the ATS satellite. Additional further analysis is required to determine the exact value of these methods when compared to the off-grid and Cowell 11^{th}-order algorithms.

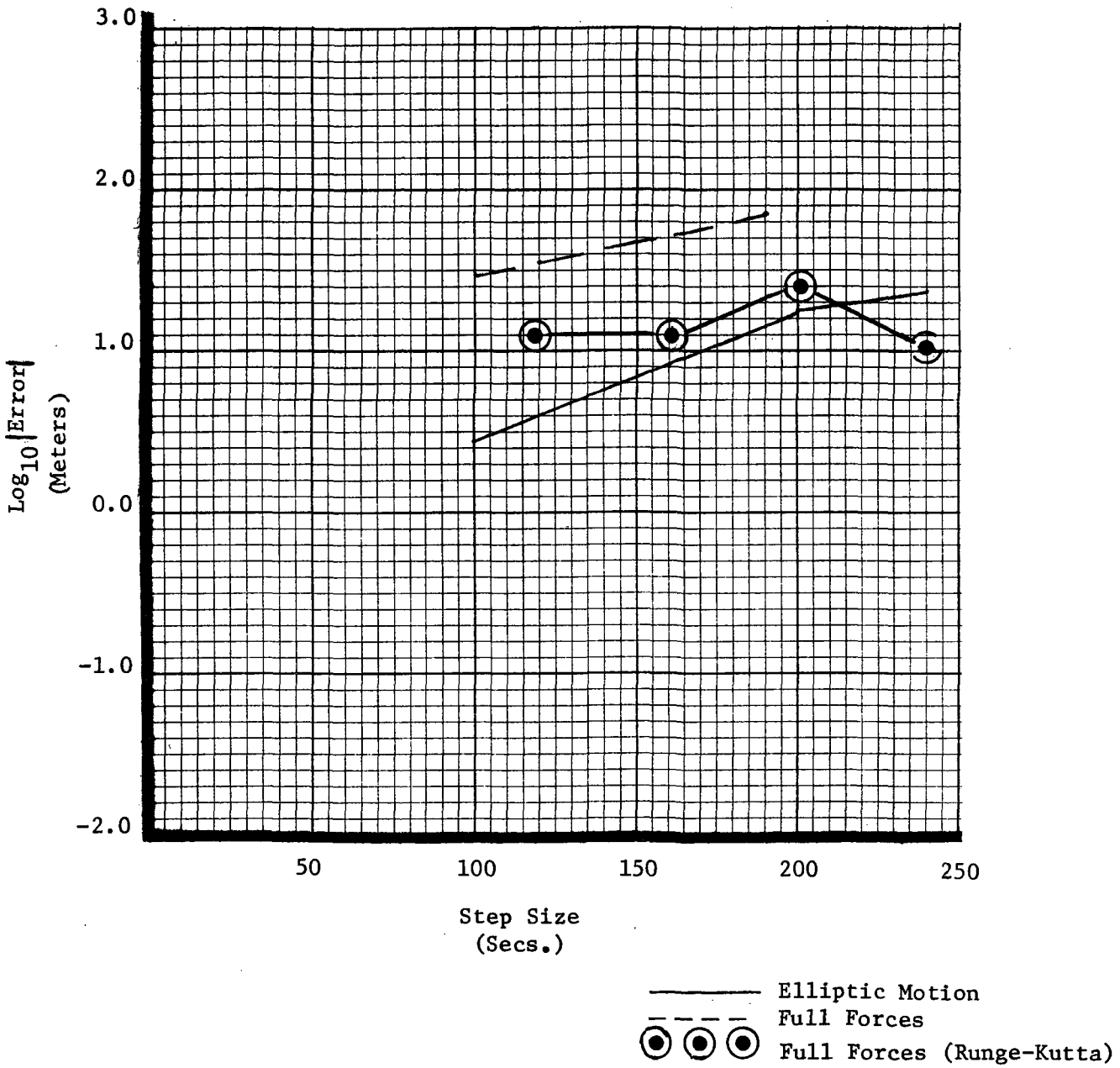

Figure 1. GEOS B Satellite--30-Day Arc
Error vs. Step Size, Off-Grid K=6

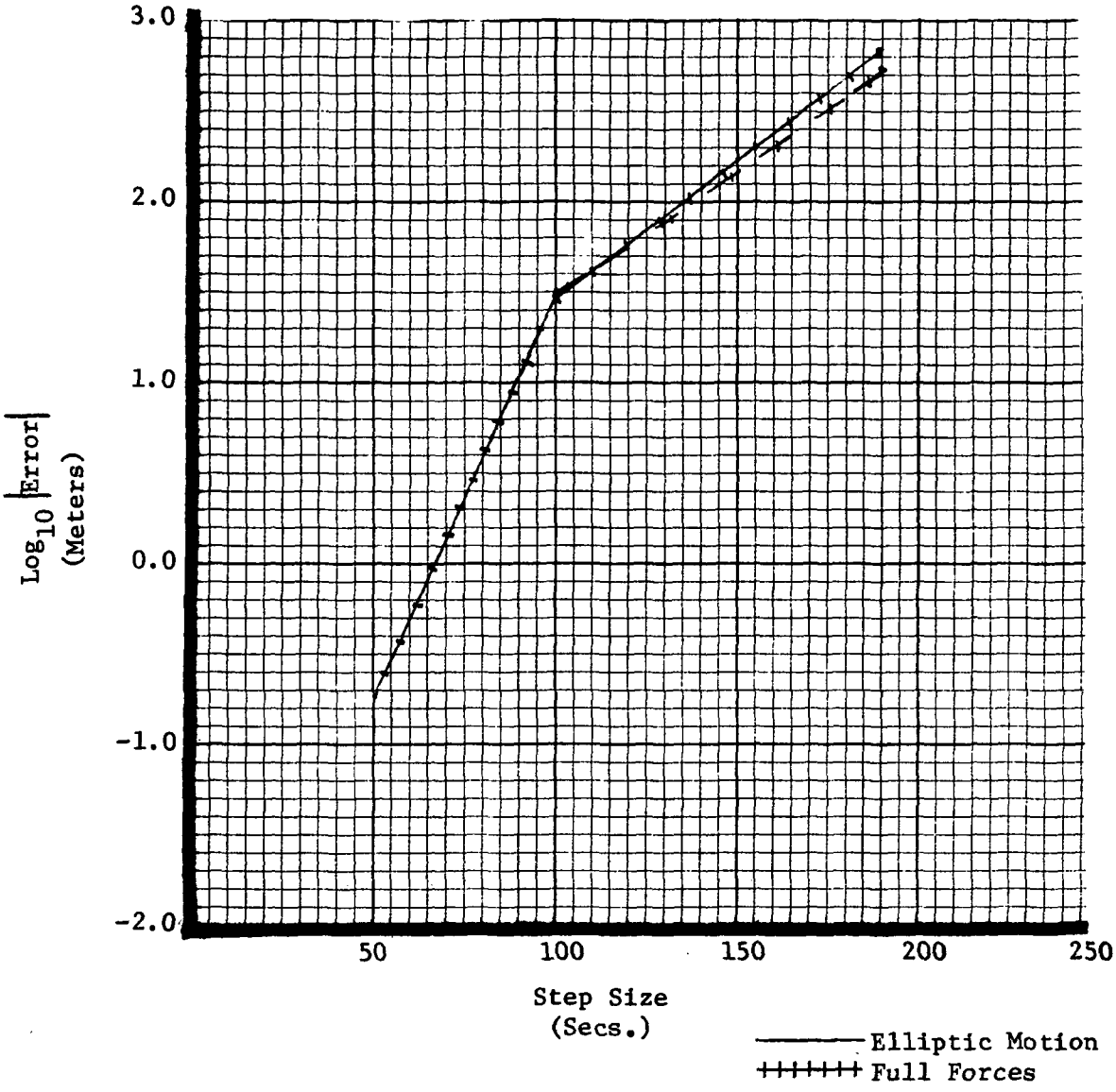

Figure 2. GEOS B Satellite--30-Day Arc
Error vs. Step Size, Cowell 11th Order

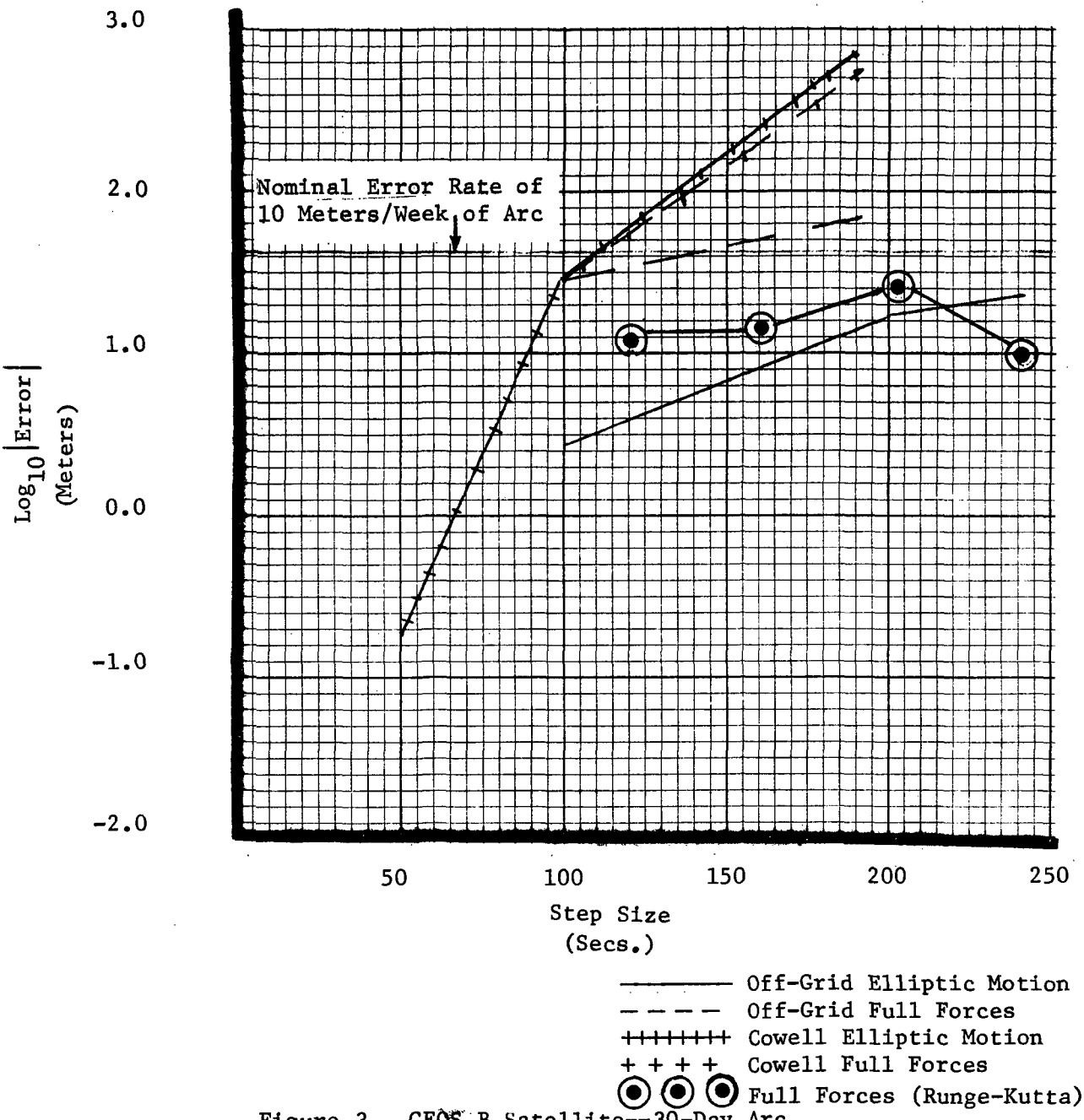

Figure 3. GEOS B Satellite--30-Day Arc
Error vs. Step Size, Cowell 11th Order, Off-Grid K=6

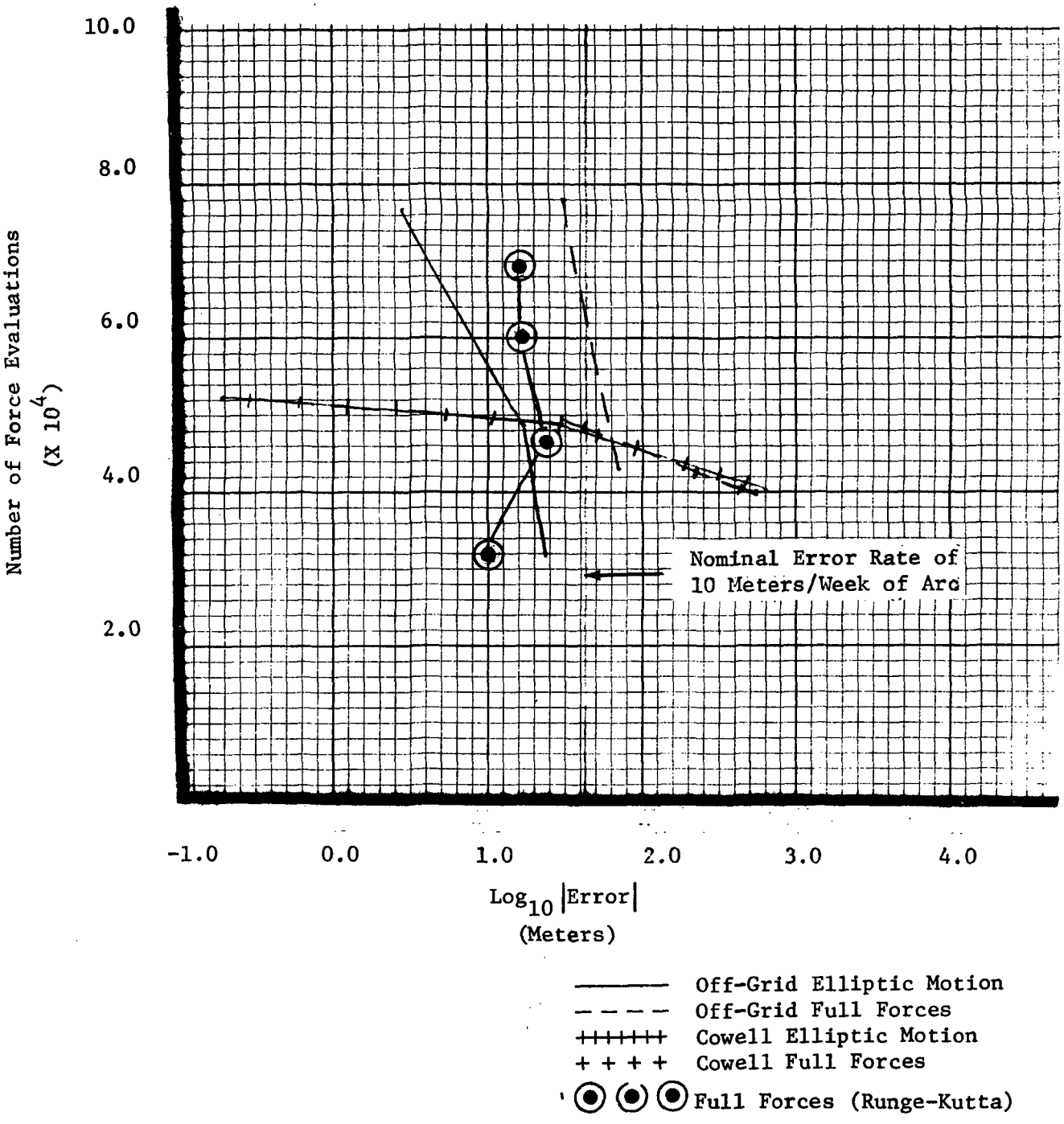

Figure 4: GEOS B Satellite--30-Day Arc
Error vs. Step Size, Cowell 11th Order, Off-Grid K=6

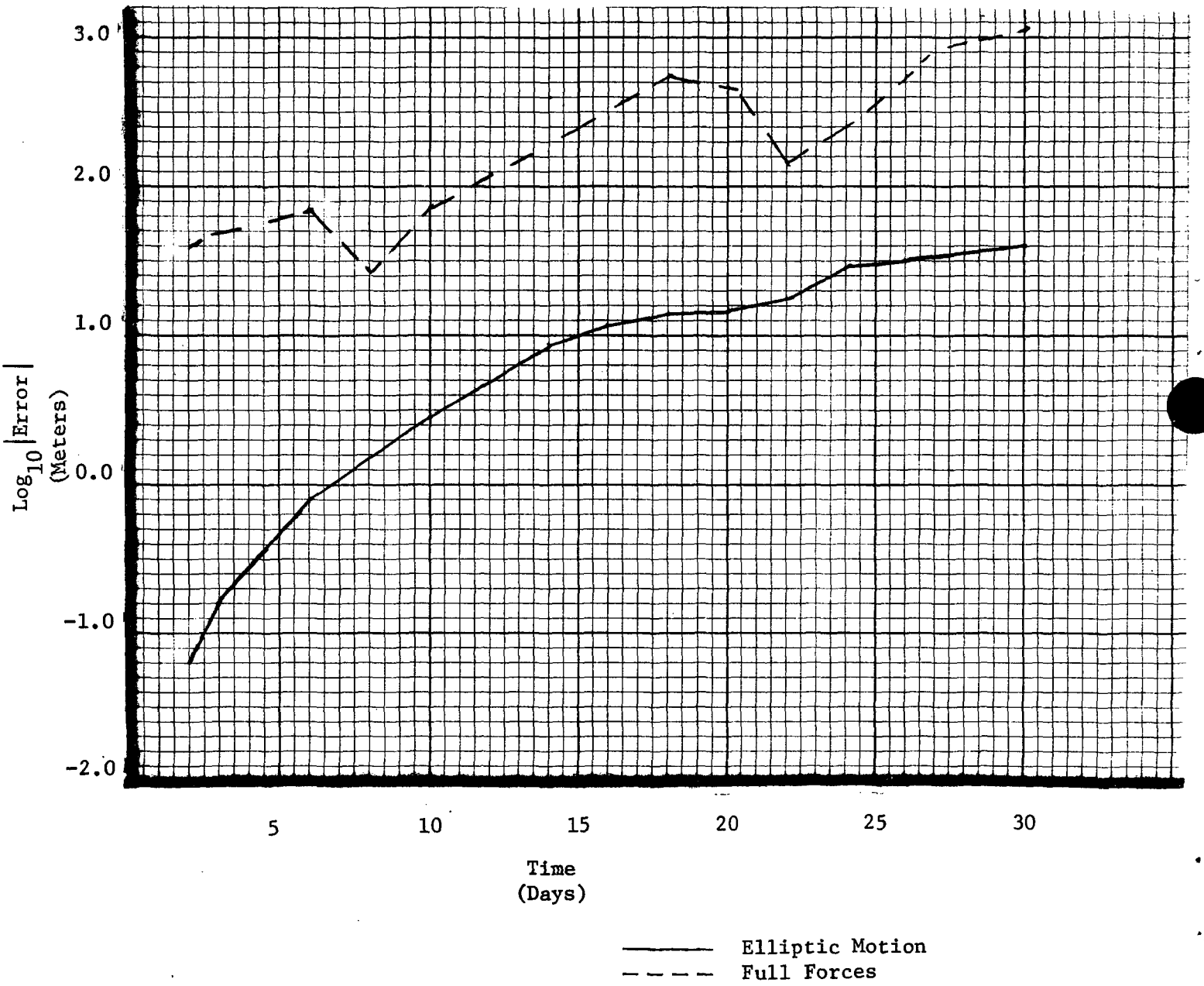

Figure 5. GEOS B Satellite
Off-Grid Propagated Error vs. Time
K=6, Step Size=240 Seconds

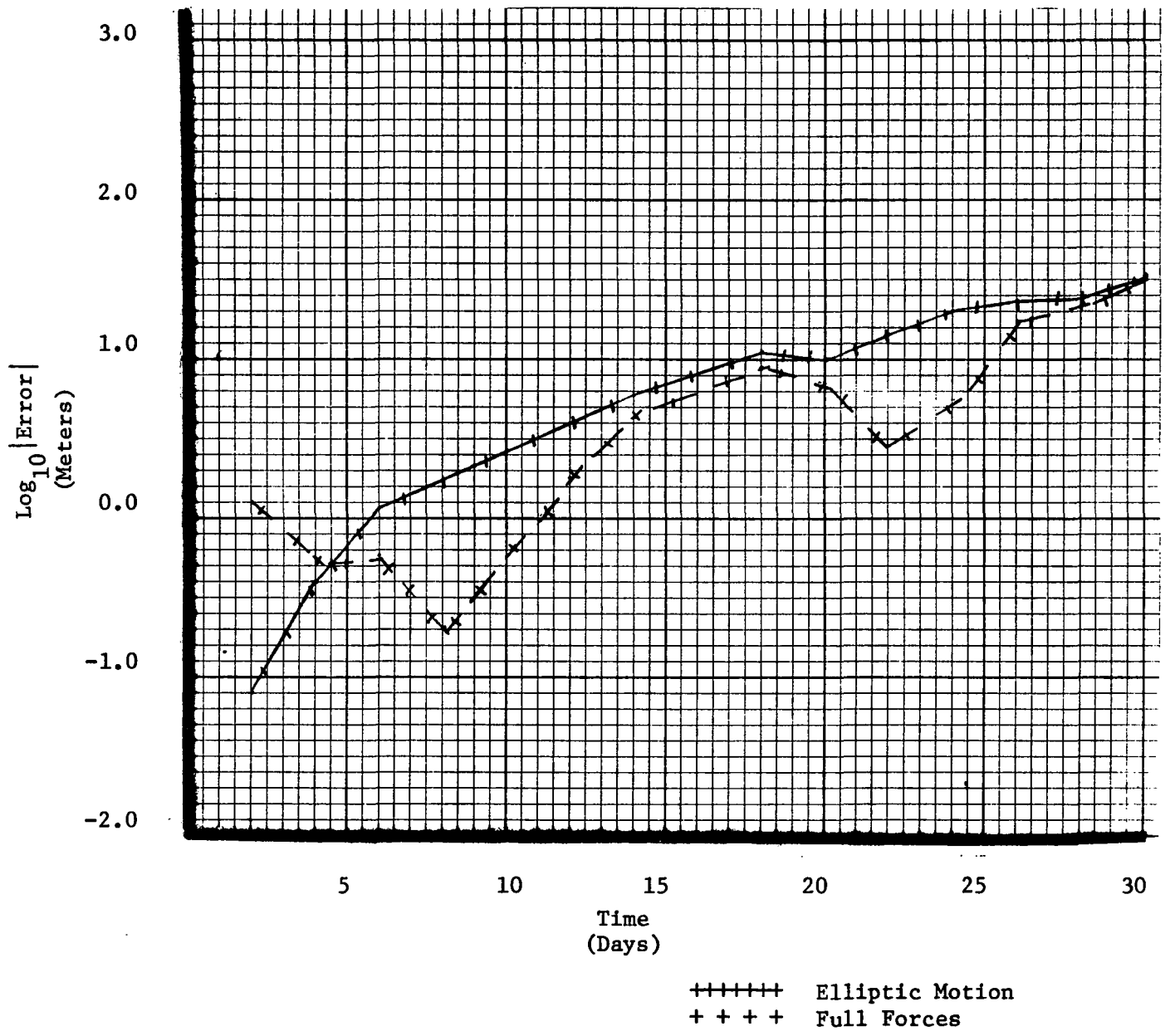

Figure 6. GEOS B Satellite
Cowell Propagated Error vs. Time
11th Order, Step Size=100 Seconds

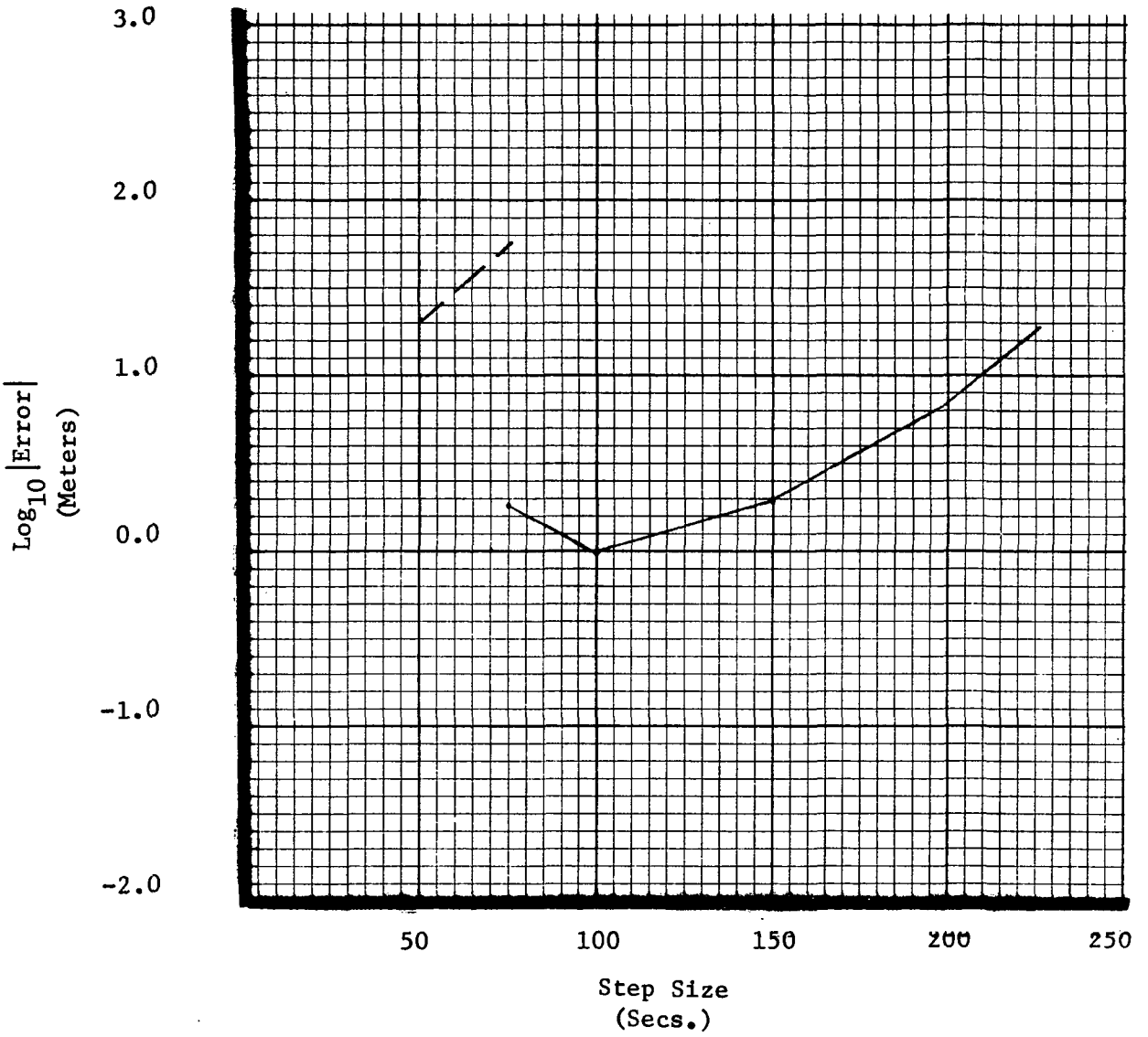

Figure 7. DELTA PAC Satellite--30-Day Arc
Error vs. Step Size, Off-Grid K=6

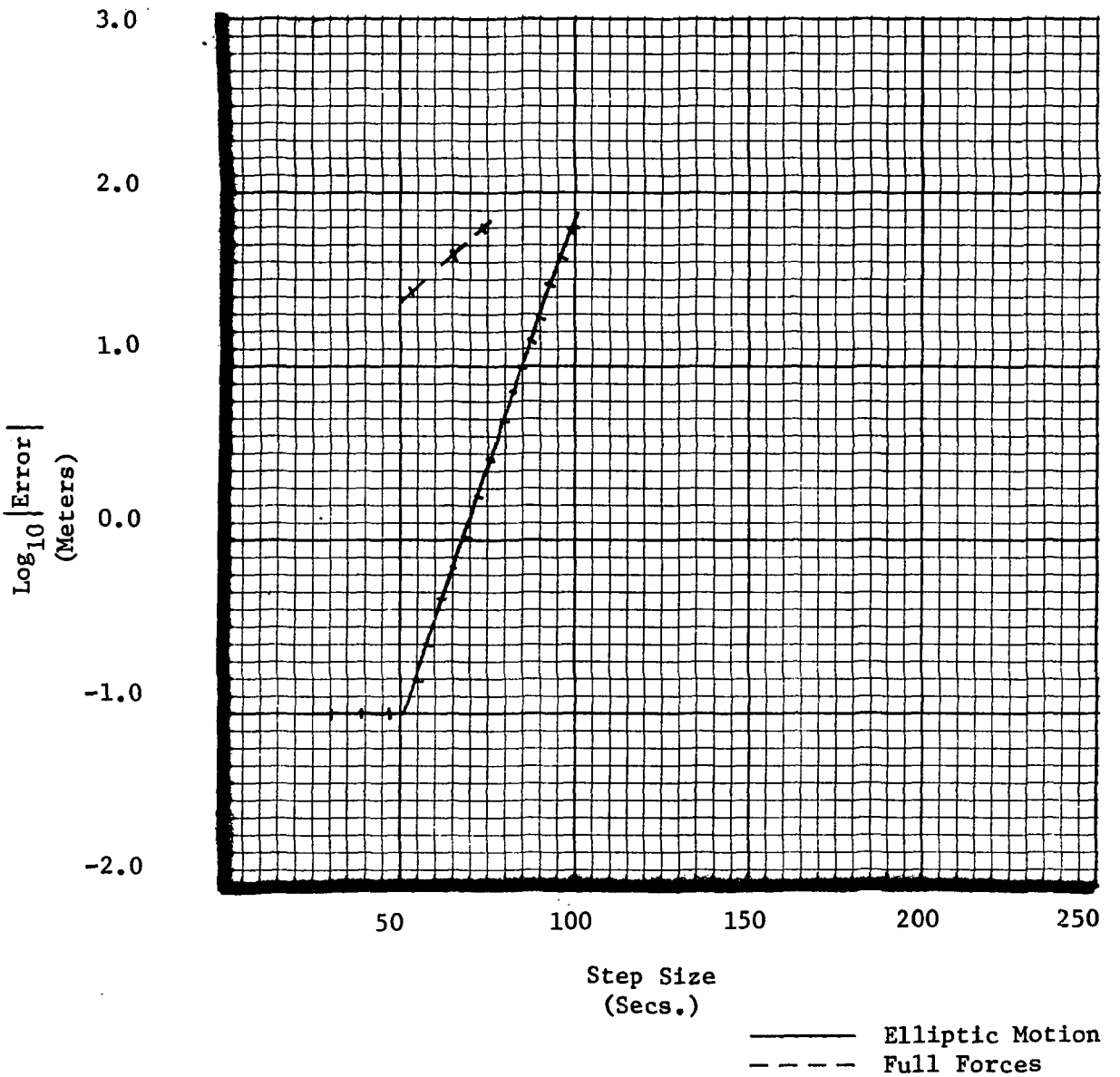

Figure 8. DELTA PAC Satellite--30-Day Arc Error vs. Step Size, Cowell 11th Order

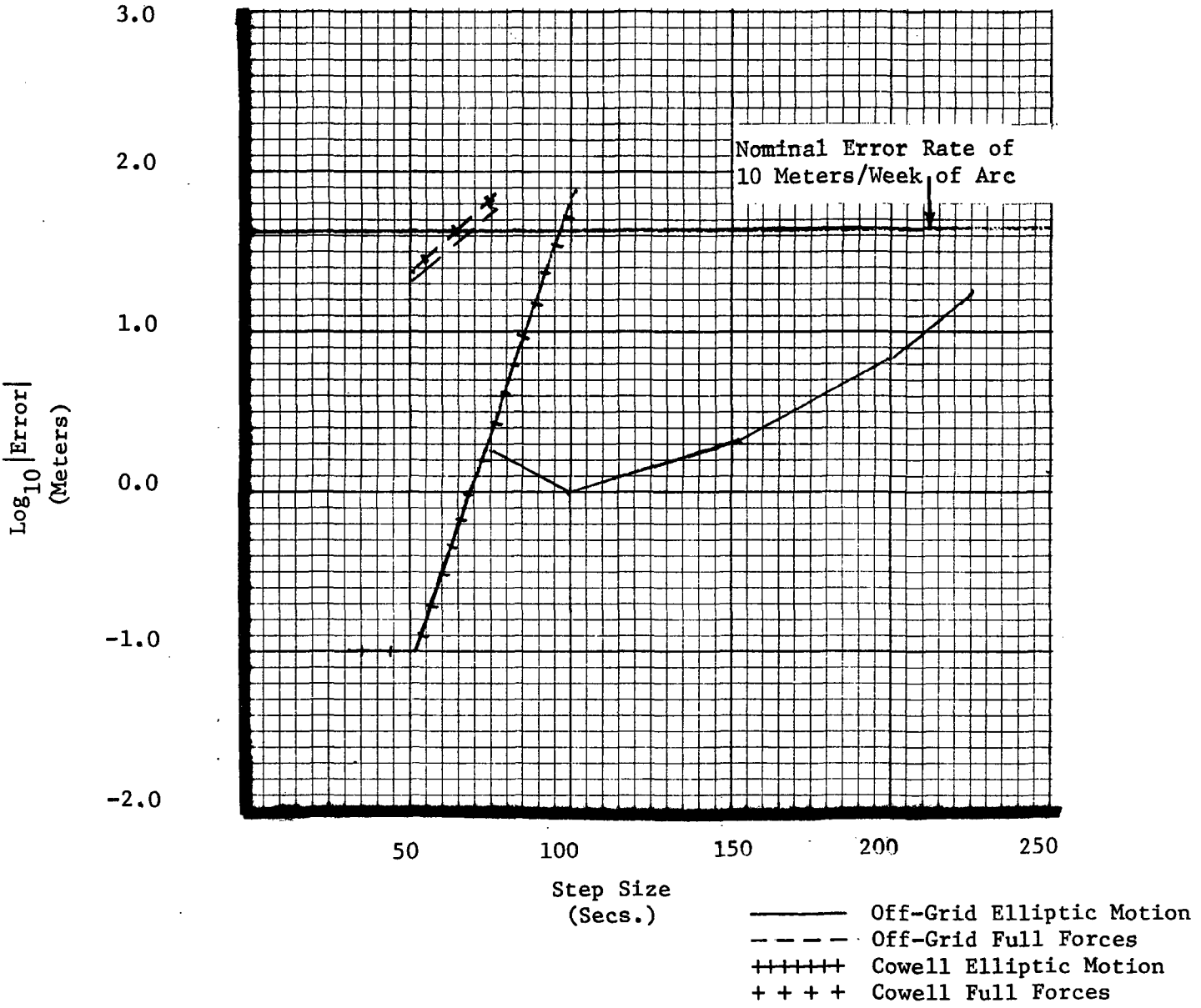

Figure 9. DELTA PAC Satellite--30-Day Arc
Error vs. Step Size, Cowell 11th Order, Off-Grid K=6

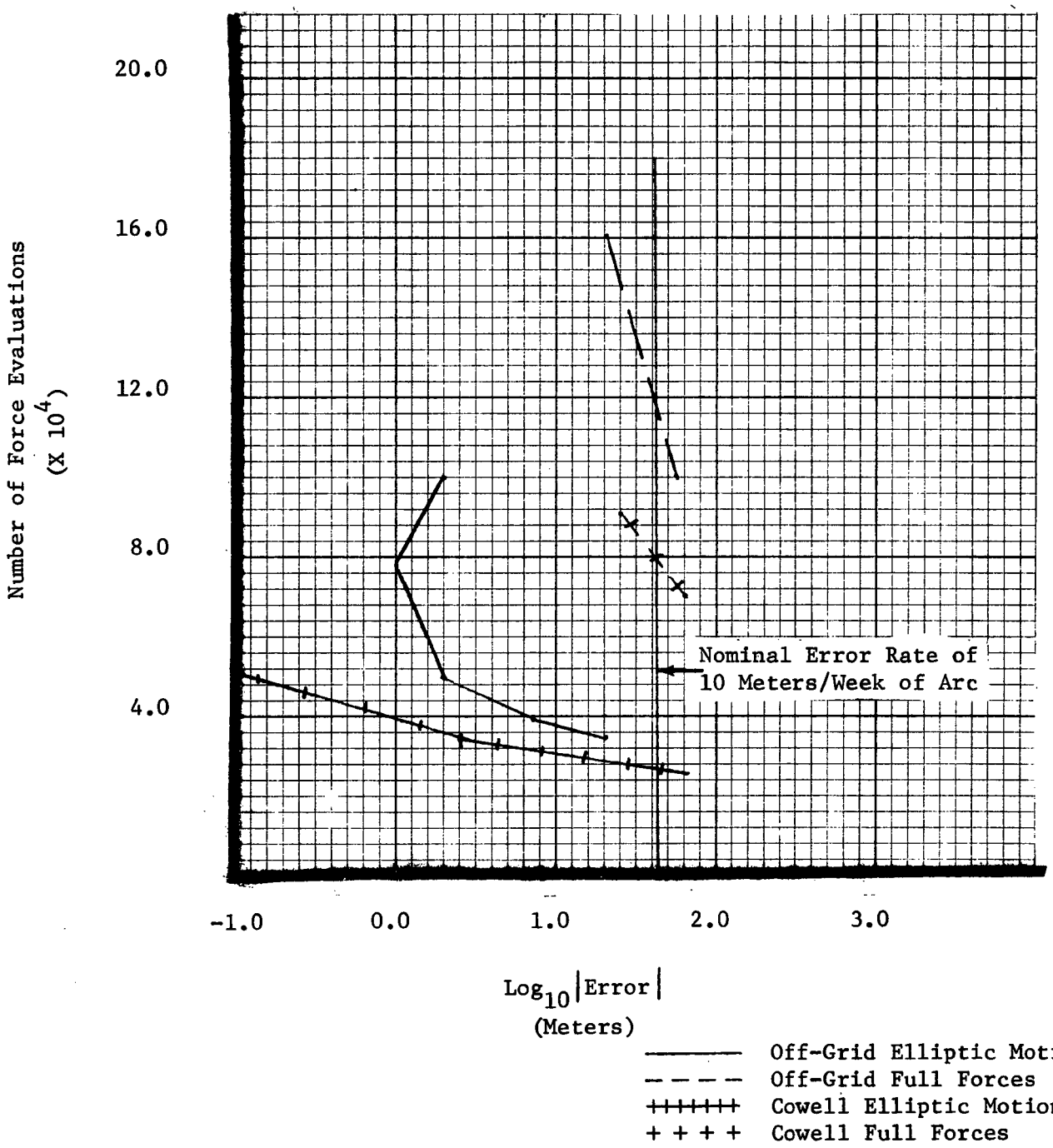

Figure 10. DELTA PAC Satellite--30-Day Arc
Number of Force Evaluations vs. Error
Cowell 11th Order, Off-Grid K=6

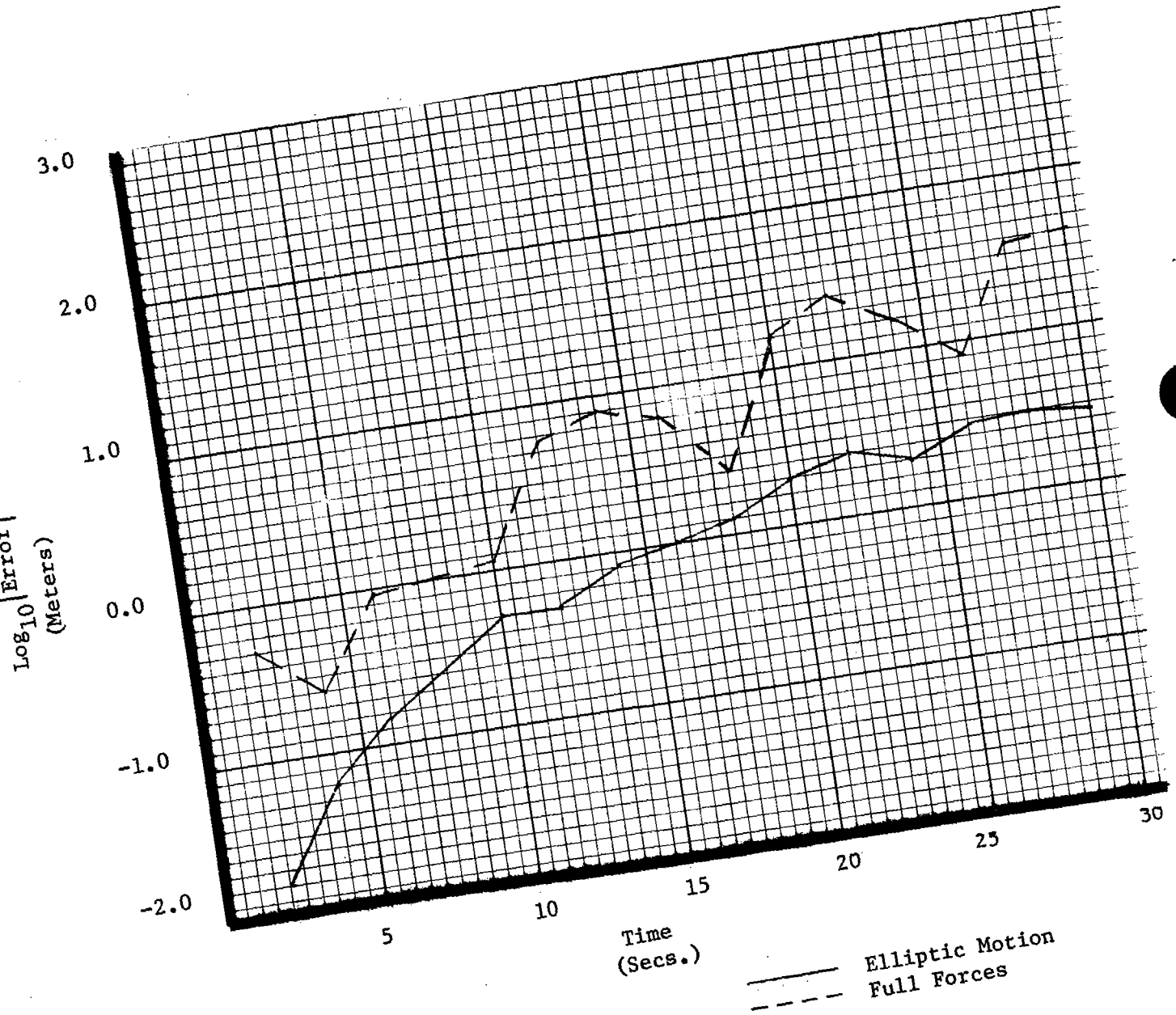

Figure 11. DELTA PAC Satellite
Off-Grid Propagated Error vs. Time
K=6, Step Size=75 Seconds

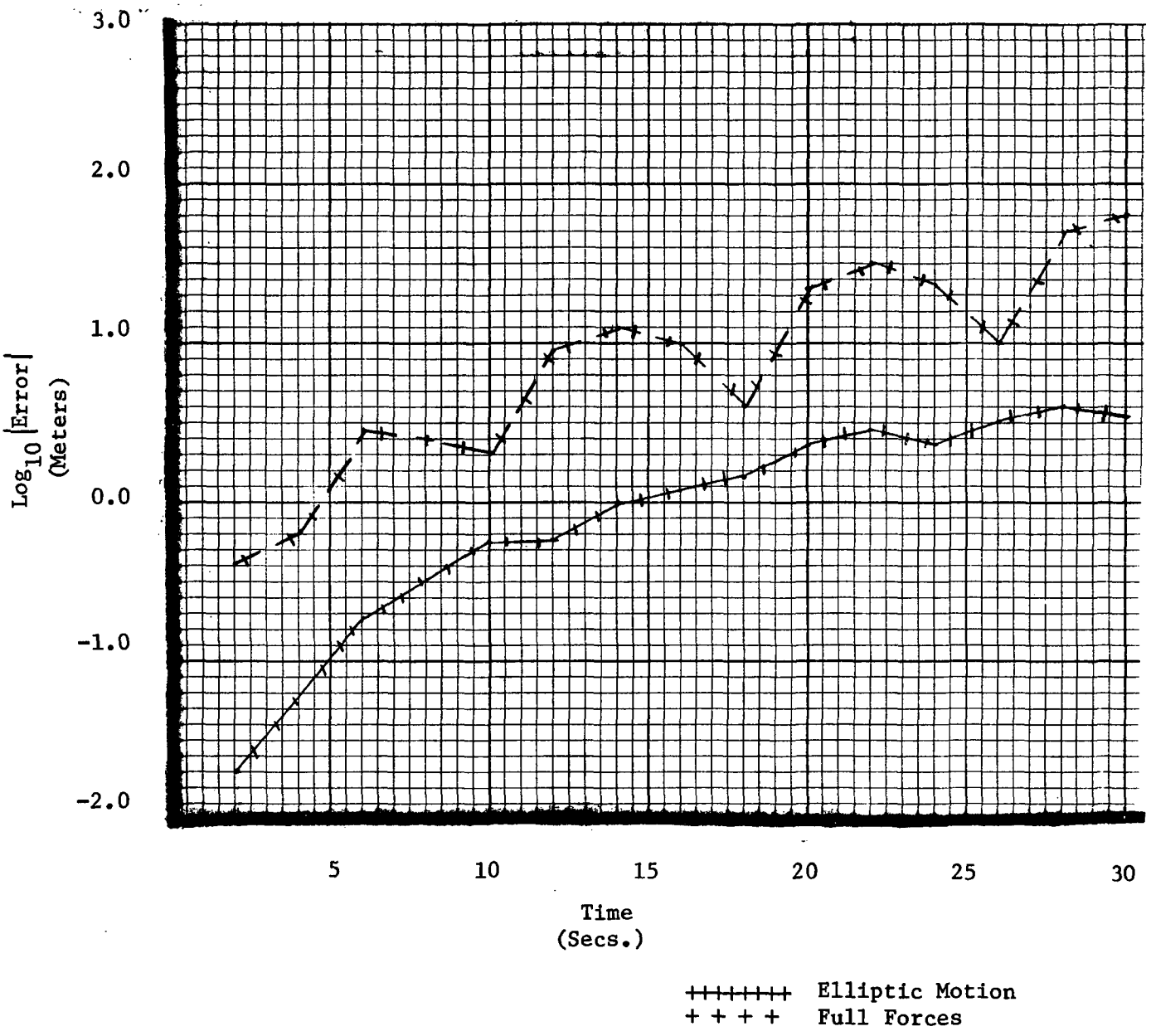

Figure 12. DELTA PAC Satellite
Cowell Propagated Error vs. Time
11th Order, Step Size=75 Seconds

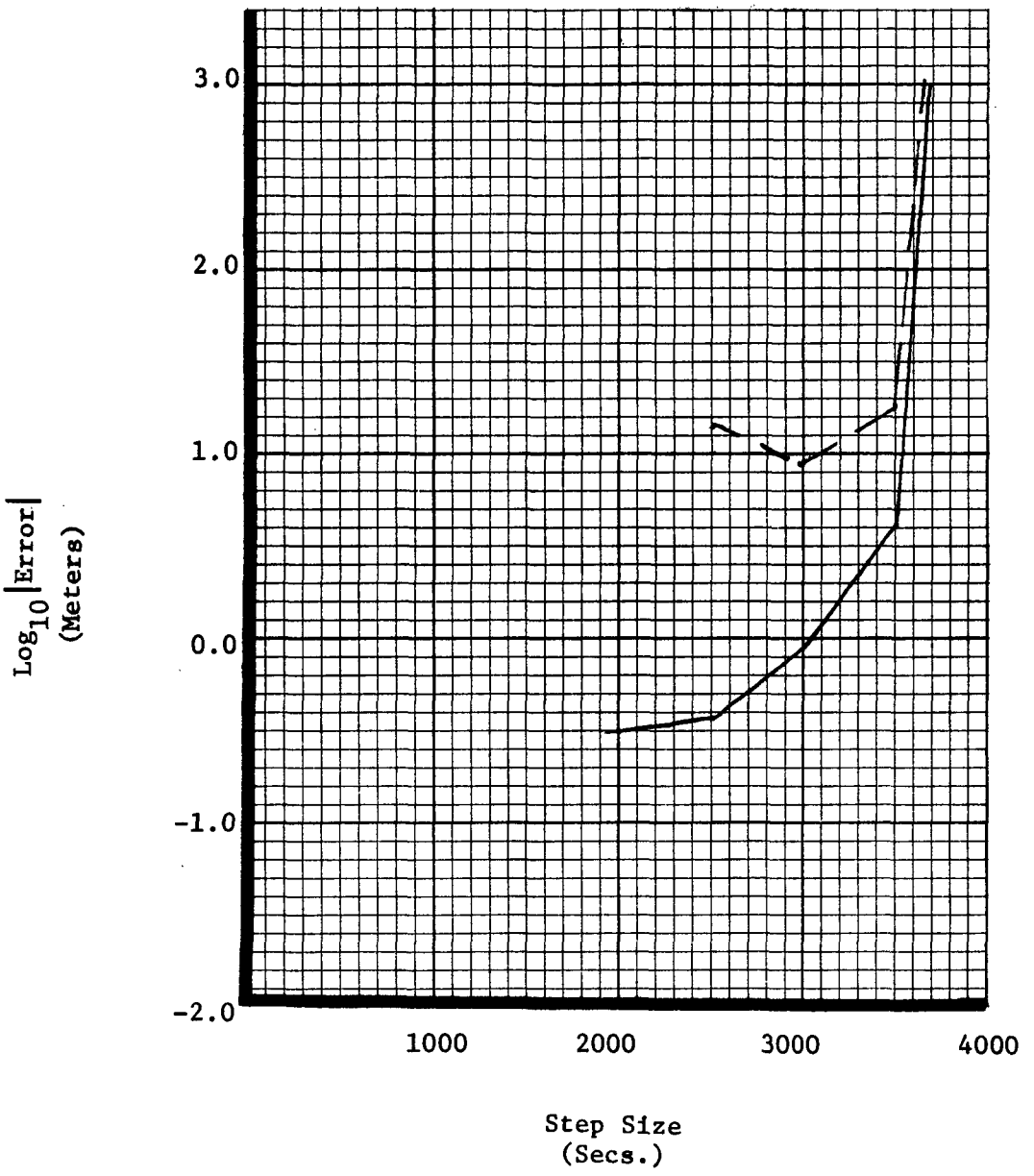

Figure 13. ATS Satellite--60-Day Arc
Error vs. Step Size, Off-Grid K=6

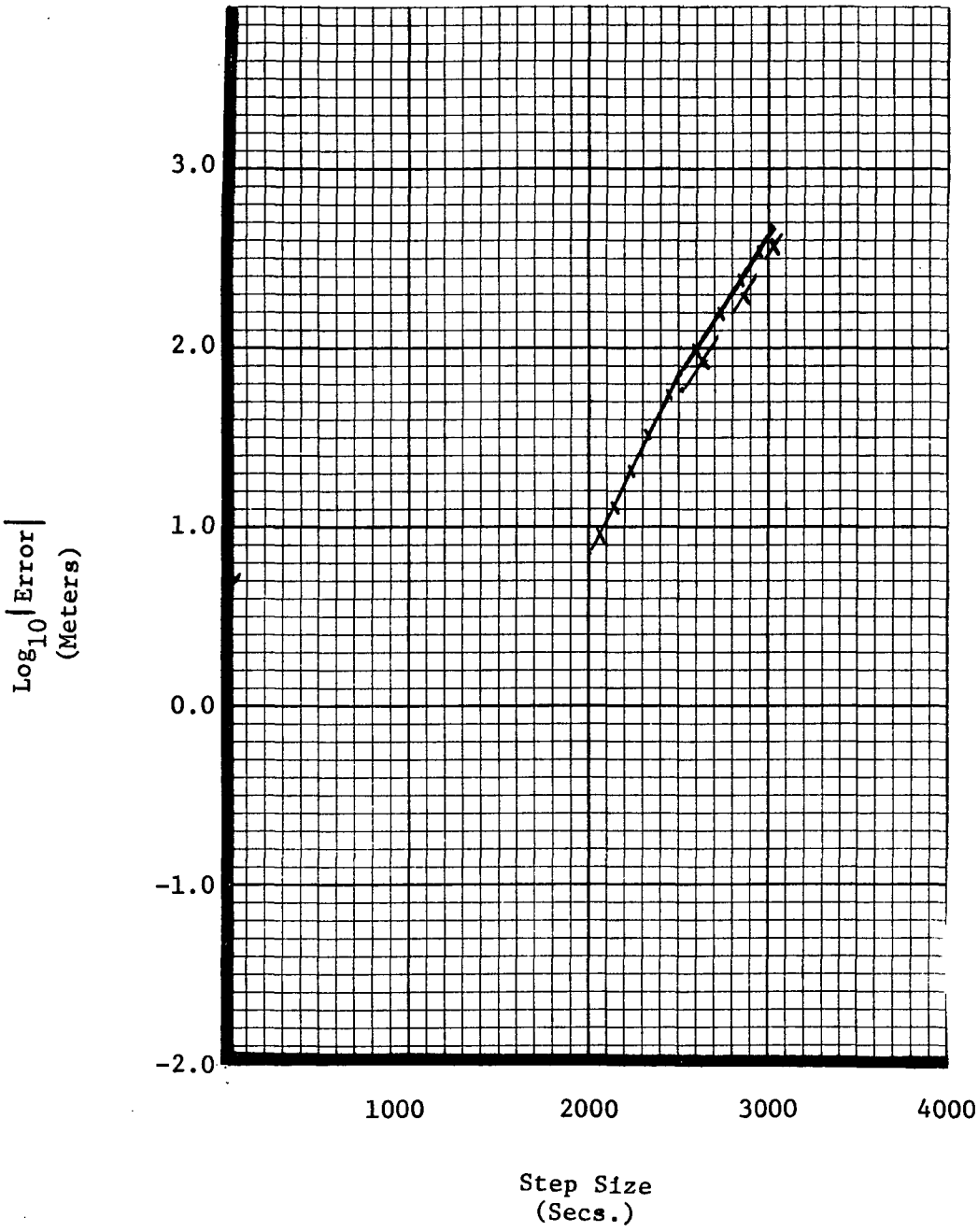

Figure 14. ATS Satellite--60-Day Arc
Error vs. Step Size, Cowell 11th Order

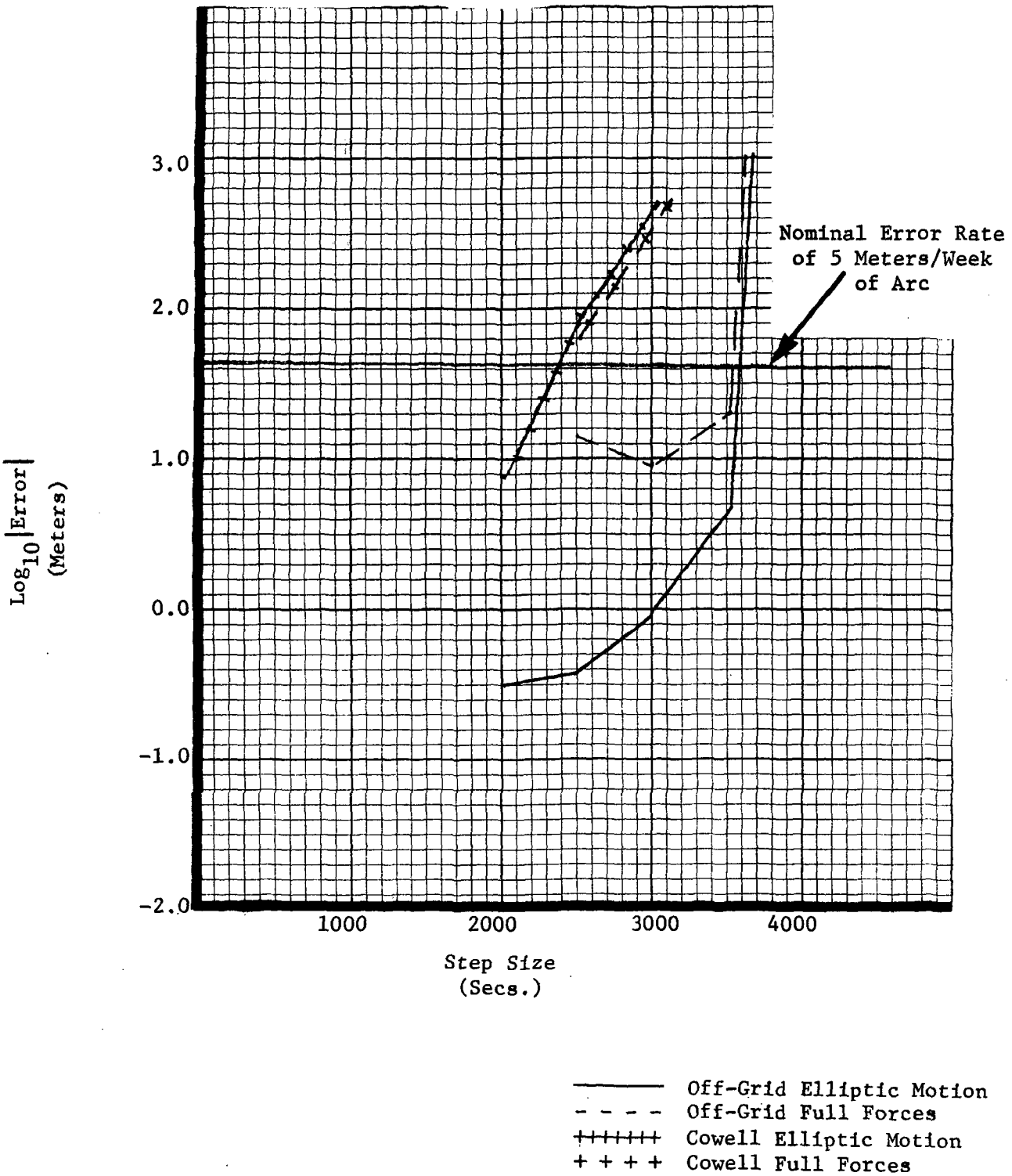

Figure 15. ATS Satellite--60-Day Arc
Error vs. Step Size, Cowell 11th Order, Off-Grid K=6

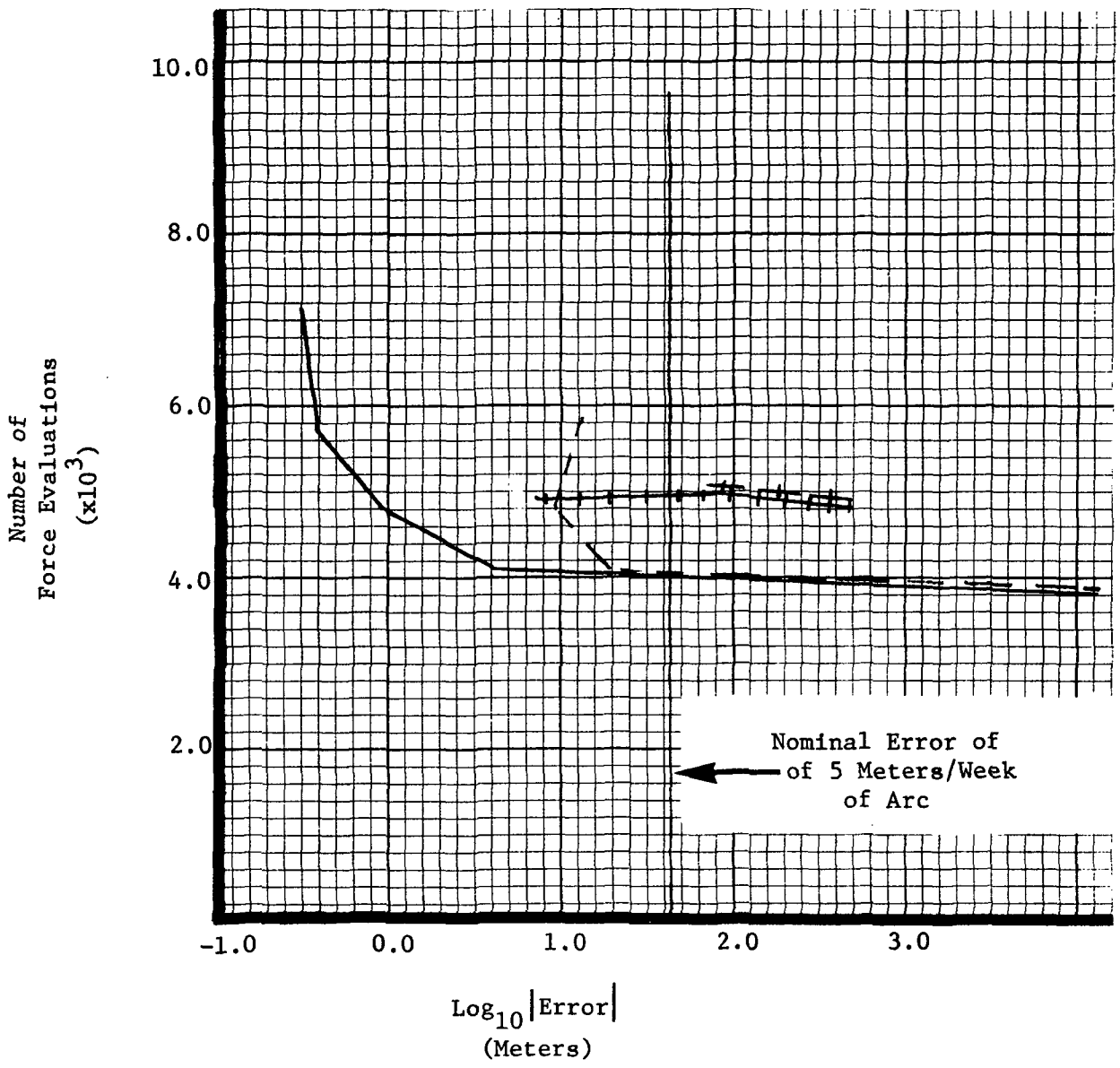

Figure 16. ATS Satellite--60-Day Arc
Number of Force Evaluations vs. Error
Cowell 11th Order, Off-Grid K=6

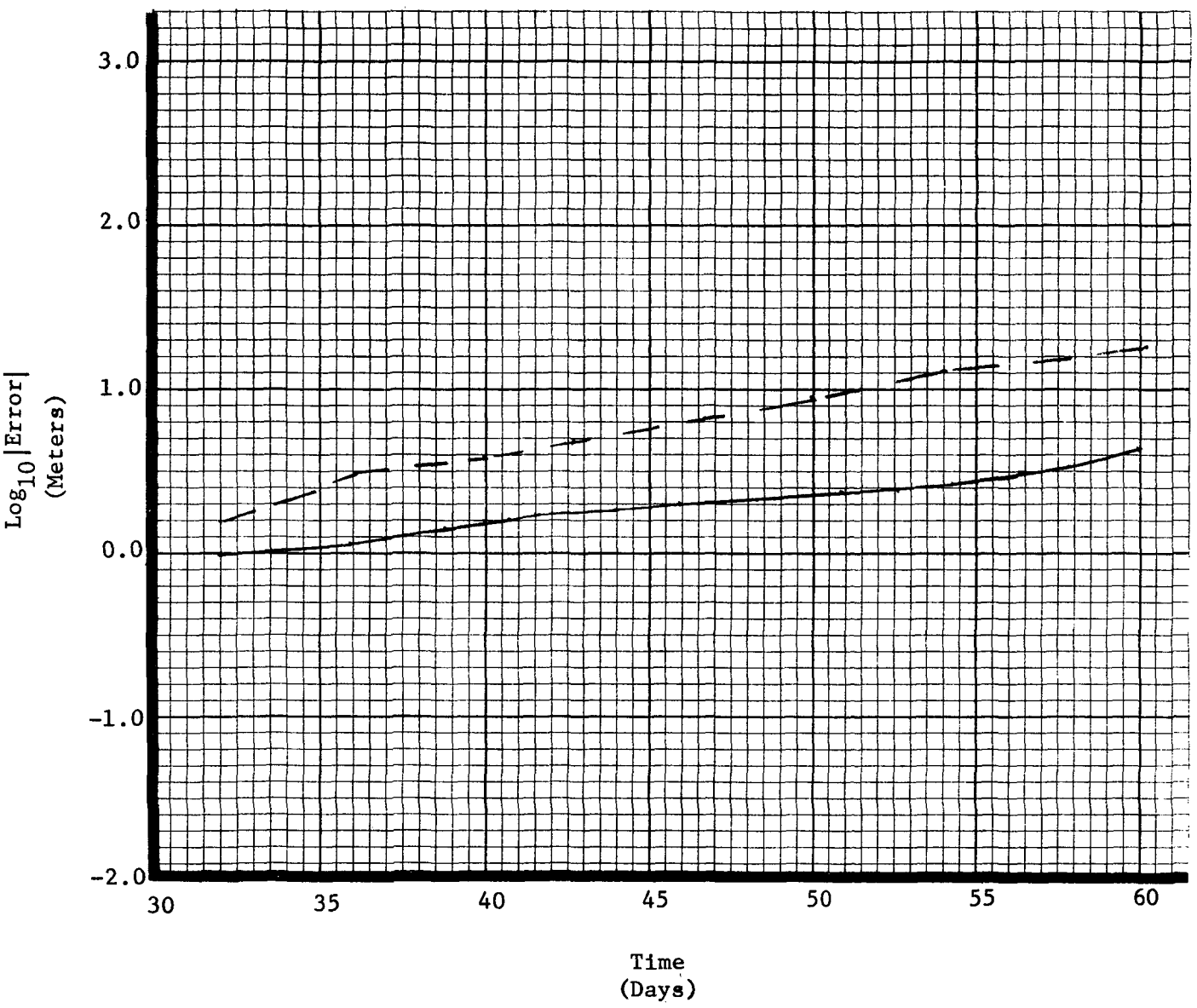

Figure 17. ATS Satellite
Off-Grid Propagated Error vs. Time
K=6, Step Size=3500 Seconds

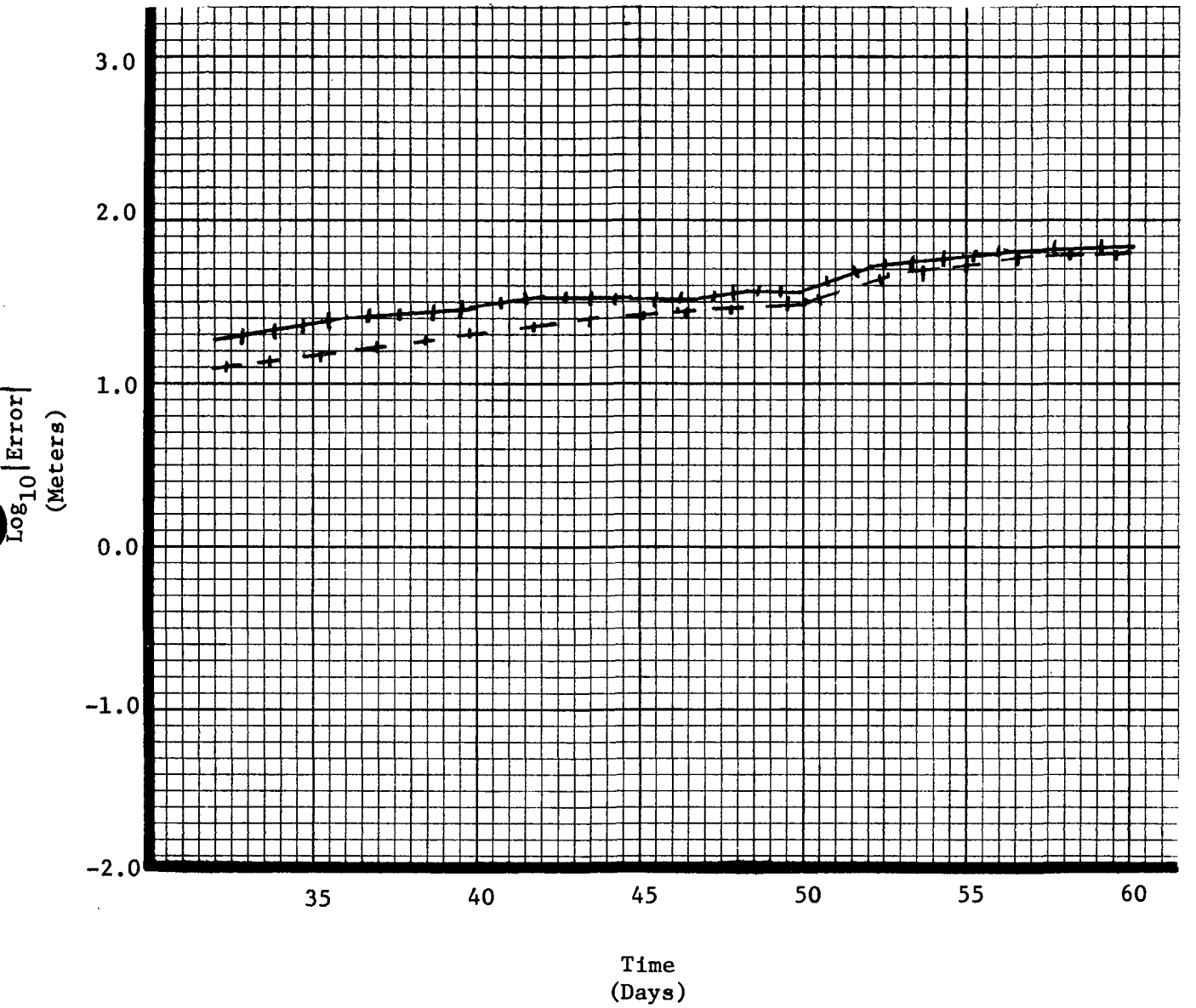

Figure 18. ATS Satellite
Cowell Propagated Error vs. Time
11th Order, Step Size=2500 Seconds

2.4 INTERPOLATION OF OFF-GRID DERIVATIVES

From a computational point of view the extra derivative evaluation required at the off-grid point is a big disadvantage for off-grid methods. Thus a method has been developed where this derivative is interpolated rather than evaluated as in the standard approach. The method was implemented on the GEOSTAR program and evaluated. Preliminary results are very encouraging. Indications are that the method achieves levels of accuracy similar to those of the present method thus allowing for greater computational efficiency.

The present method uses a 9^{th}-degree Lagrange interpolating polynomial fit through nine back points. Possibly the method could be further improved by either using additional back points to achieve a higher-degree polynomial, or by using, if feasible, a quasi-Hermite interpolating technique.

In the standard off-grid method the values of y_n, y_{n-1}, -- y_{n-k+1} are used to calculate the value of y_{n+1}. First the predicted values of $y^{(0)}_{n+1}$ and $y^{(0)}_{n+1-\theta}$ are calculated. Then the second derivatives $\ddot{y}_{n+1} = f\left(y^{(0)}_{n+1}\right)$ and $\ddot{y}_{n+1-\theta} = f\left(y^{(0)}_{n+1-\theta}\right)$ are evaluated. The corrected value of $y^{(1)}_{n+1}$ is calculated and, finally, the corrected value of $\ddot{y}_{n+1} = f\left(y^{(1)}_{n+1}\right)$ is evaluated. This procedure involves three force evaluations.

In the modified procedure the predicted values of $y^{(0)}_{n+1}$ and $y^{(0)}_{n+1-\theta}$ are calculated just as before. The force function however is broken down into two components. A two-body component and a perturbing-forces component. Both components are evaluated at $y^{(0)}_{n+1}$ while only the two-body component is evaluated at $y^{(0)}_{n+1-\theta}$. The perturbed component at the off-grid point is then interpolated using a 9^{th}-degree Lagrange polynomial fit through the perturbed component of $f\left(y^{(0)}_{n+1}\right)$, and nine previous points. Stability of the method is ensured by use of the corrector equation. The only issue is the accuracy of the interpolated point in comparison to the rest of the method. Results to date indicate that the method is stable and almost as accurate as the standard k=6 method.

2.5 VARIABLE STEP PROCEDURES

2.5.1 Generalization of Milne's Method

A variable-step procedure for use with off-grid methods has been developed. This procedure allows for the automatic variation of the integration step size. It is based on a generalization of Milne's method [8] for estimating local discretization errors. The approach generalizes that previously implemented in the GEOSTAR program [7]. The newly developed methods have been implemented in the GEOSTAR program and, after being properly calibrated, can be used to evaluate the effectiveness of off-grid methods for highly eccentric orbits.

The results are applicable to most multistep integration methods including off-grid methods. In the case of the Cowell methods, they reduce to the presently used method. The present discussion is based on integration methods for solving first-order differential equations. An analogous procedure has been used for the second-order case.

A large class of multistep methods can be expressed in the operator form

$$L(x,y(x),h) = a_k(x+kh) + a_{k-1}y(x+(k-1)h) \text{ --- } a_o y(x) + h\{b_k y'(x+kh)+b_{k-1}y'(x+(k-1)h) \text{ --- } b_o y'(x)\}. \quad (2\text{-}7)$$

Such methods are used to solve the initial-value problem

$$y' = f(x,y),\ y'(a) = C. \quad (2\text{-}8)$$

In this case $y'(x+kh) = f(x+kh,\ y(x+kh))$ and $y(x+kh)$ are derived from expression (2-7) given an appropriate set of starting values. The integration step size is h, and k is the number of back points used in the approximation procedure.

If method (2-7) is of order P and if $z(x+kh)$ is the true solution to (2-8) then (2-7) becomes

$$L(x,z(x),h) = C_{P+1}h^{P+1}(P+1)\ z(x) + 0(h^{P+2}) \quad (2\text{-}9)$$

where

$$z^{(P+1)}(x) = \left.\frac{d^{P+1} z(r)}{dz(r)^{P+1}}\right|_{r=x} \tag{2-10}$$

and

$$C_{P+1} = \frac{\left(a_1 + 2^{P+1} a_2 \text{ --- } k^{P+1} a_k\right)}{(P+1)!} + \frac{\left(b_1 + 2^{P} b_2 \text{ --- } k^{P} b_k\right)}{P!} . \tag{2-11}$$

In most cases of application, a predictor (i.e. (2-7) with $b_k=0$) is first used to approximate $y(x+kh)$ and then a corrector (2-7) with $b_k \neq 0$) is used to obtain the final solution for $y(x+kh)$. If the predictor and corrector are of the same order P, then they can be written for $z(x+kh)$ as

Predictor

$$\begin{aligned} a_k^* z(x+kh) \text{ ----- } a_o z(x) + h\left\{b_{k-1} z'(x+kh) \text{ ---- } b_o z'(x)\right\} \\ + C_{P+1}^* h^{P+1} z^{(P+1)}(x) + 0(h^{P+2}) = 0 \end{aligned} \tag{2-12}$$

Corrector

$$\begin{aligned} a_k z(x+kh) \text{ ----- } a_o z(x) + h\left\{b_k z'(x+kh) \text{ ---- } b_o z'(x)\right\} \\ + C_{P+1} h^{P+1} z^{(P+1)}(x) + 0(h^{P+2}) = 0 \end{aligned} \tag{2-13}$$

If we define local discretization error for the corrector and predictor as

Predictor

$$R_{m+k}^* = z(x_m+kh) - y_{m+k}^* = \epsilon_{m+k}^* \tag{2-14}$$

Corrector

$$R_{m+k} = z(x_m+kh) - y_{m+k} = \epsilon_{m+k} \tag{2-15}$$

where y^*_{m+k} and y_{m+k} are the solutions to the predictor and corrector equations, then expressions for ϵ_{m+k} and ϵ^*_{m+k} are given by

Predictor

$$\begin{aligned} & a^*_K \epsilon^*_{m+k} + \text{-- } a^*_o \epsilon^*_m \\ & + h\left\{b^*_{k-1}(z'(x_m+(k-1)h) - y'_{m+k-1}) \text{ ---- } b^*_o(z'(x_m) - y'm)\right\} \\ & + C^*_{P+1} h^{P+1} z^{(P+1)}_{(x_m)} + O(h^{P+2}) \end{aligned} \tag{2-16}$$

Corrector

$$\begin{aligned} & a_k \epsilon_{m+k} \text{ --- } a_o \epsilon_m \\ & + h\left\{b_k(z'(x_m+kh) - y'_{m+k}) \text{ --- } b_o(z'(x_m) - y'_m)\right\} \\ & + C_{P+1} h^{P+1} z^{(P+1)}_{(x_m)} + O(h^{P+2}) \end{aligned} \tag{2-17}$$

From the mean value theorem

$$z'(x) - y'(x) = \frac{\partial f(x,\Theta)}{\partial z}(z(x) - y(x)) \tag{2-18}$$

Thus the terms $\left\{ \quad \right\}$ in (2-16) and (2-17) are of order $O(h^{P+2})$. Using this result, setting $a^*_k = a_k = 1$ and subtracting (2-16) from (2-17) we get

$$(y_{m+k}-y^*_{m+k}) + a_{k-1}\epsilon_{m+k-1}-a^*_{k-1}\epsilon^*_{m+k-1} \text{ --- } a_o\epsilon_m-a^*_o\epsilon^*_m + h^{P+1}z^{(P+1)}_{(x_m)}\ [C_{P+1}-C^*_{P+1}] + 0(h^{P+2}) = 0 \tag{2-19}$$

or

$$z^{(P+1)}_{(x_m)} = \frac{(y_{m+k}-y^*_{m+k})+R}{(C^*_{P+1}-C_{P+1})h^{P+1}} + 0(h^{P+2}) \tag{2-20}$$

where

$$R = a_{k-1}\epsilon_{m+k-1}-a^*_{k-1}\epsilon^*_{m+k-1} \text{ ---- } a_o\epsilon_m-a^*_o\epsilon_m \tag{2-21}$$

If the first k starting values are exact relative to the required accuracy (i.e., $\epsilon_m, \epsilon^*_m = 0(h^{P+2})$ for m=0,1, --, k-1) then the error terms are calculated recursively by:

$$\epsilon_m = \left[\frac{(y_m-y^*_m)+R}{(C^*_{P+1}-C_{P+1})}\right] C_{P+1} + 0(h)^{P+2} \tag{2-22}$$

$$\epsilon^*_m = \left[\frac{(y_m-y^*_m)+R}{C^*_{P+1}-C_{P+1}}\right] C^*_{P+1} + 0(h)^{P+2} \tag{2-23}$$

$$\epsilon_{k+1} = \frac{(y_{k+1}-y^*_{k+1})}{(C^*_{P+1}-C_{P+1})} C_{P+1} + 0(h^{P+2}) \tag{2-24}$$

$$\epsilon^{*}_{k+1} = (\epsilon_{k+1}) \frac{C^{*}_{P+1}}{C_{P+1}} + O(h^{P+2}) \qquad (2\text{-}25)$$

2.5.2 Application to Off-Grid Methods

In a manner analogous to the previous development, the off-grid methods are represented for Class I by

$$L[x,y(x),h,\theta] = a_k y(x+kh) \text{ ---- } {}_o y(x)$$

$$+ h\{b_k y'(x+kh) + b_\theta y'(x+(k-\theta)h) \text{ ---- } b_o y'(x)\} \qquad (2\text{-}26)$$

If $z(x+kh)$ is the true solution to the differential equation and if the method is exact for a polynomial of degree P then

$$L[x,z(x),h,\theta] = C_{P+1} h^{P+1(P+1)} z(x) + O(h^{P+2}) \qquad (2\text{-}27)$$

with

$$C_{P+1} = \frac{(a_1+2^{P+1}a_2 \text{ ---- } ka_k^{P+1})}{(P+1)!} + \frac{(b_1+2^{P}b_2 \text{ --- } k^{P}b_k+(k-\theta)^{P}b_\theta)}{P!} \; . \qquad (2\text{-}28)$$

The rest of the analysis is similar to that in the previous section. For Class II equations the value of P is replaced by P-1 in the right hand term of (2-28), i.e., the term with the b's.

To achieve the necessary order the off-grid predictors have one more back point than the correctors (e.g. the k=6 method uses a 7-point predictor, etc.). However, for the variable-step procedure both methods must have the same value of k. This is remedied by calling the corrector a k+1 step method with the value of a_0 and $b_0 = 0$. C_{P+1} in (2-28) then must be calculated using k+1 instead of k. Calling this new value C^*_{P+1}, it can be shown for Class II that in general

$$C^*_P = C_P + C_{P-1} + \frac{C_{P-2}}{2!} + \frac{C_{P-3}}{3!} \text{ ---- } \frac{C_1}{(P-1)!} \qquad (2\text{-}29)$$

provided $C_2 = C_0 = 0$

since our methods are exact for polynomials of degree P, C_0, C_1, C_2 -- $C_P = 0$ and $C^*_{P+1} = C_{P+1}$.

2.6 COEFFICIENTS OF OFF-GRID METHODS

Using the program described in Appendix A, the following coefficients for 8^{th}, 10^{th}, and 11^{th} order (k = 4, 5, 6) methods have been developed. Coefficients for K = 6 were presented in Section 2.2.1 Coefficients for K = 4 and 5 are presented in the following paragraph. The notation is the same as that used for K = 6. For K = 4, the following computation sequence is followed.

Predict $\dot{y}_{n+1-\theta}$ (Class I Predictor)

Predict $y_{n+1-\theta}$ (Class II Predictor)

Evaluate $\ddot{y}_{n+1-\theta} = f(t_{n+1-\theta}, y_{n+1-\theta}, \dot{y}_{n+1-\theta})$

Correct $\dot{y}_{n+1}$ (Class I Corrector)

Correct y_{n+1} (Class II Corrector)

Evaluate $\ddot{y}_{n+1} = f(t_{m+1}, y_{n+1}, \dot{y}_{n+1})$,

hence requiring 2 function evolutions per step.

For K = S, the following computation sequence is used

Predict $\dot{y}_{n+1}$ (Class I Predictor)

Predict y_{n+1} (Class II Predictor)

Predict $\dot{y}_{n+1-\theta}$ (Class I Predictor)

Predict $y_{n+1-\theta}$

Evaluate $\ddot{y}_{n+1} = f(t_{n+1}, y_{n+1}, \dot{y}_{n+1})$

Evaluate $\ddot{y}_{n+1-\theta} = f(t_{n+1-\theta}, y_{n+1-\theta}, \dot{y}_{n+1-\theta})$

Correct $\dot{y}_{n+1}$ (Class I Corrector)

Correct y_{n+1} (Class II Corrector)

Evaluate $\ddot{y}_{n+1} = f(t_{n+1}, y_{n+1}, \dot{y}_{n+1})$,

hence requiring 3 function evolutions per step.

The following coefficients have been developed for these two methods.

Four Step Algorithm (8th Order) $\theta = .30$

Class I coefficients in the prediction of $\dot{y}_{n+1-\theta}$

$a_0 = -12.551710725$	$b_0 = 5.608211175$
$a_1 = -8.557858575$	$b_1 = 20.783370825$
$a_2 = 17.932428325$	$b_2 = 13.085826075$
$a_3 = 4.177140975$	$b_3 = 1.061012925$

Class II coefficients in the prediction of $y_{n+1-\theta}$

$a_0 = -1.9320993$	$b_0 = 0.862500525$
$a_1 = 6.4719549$	$b_1 = 3.030380475$
$a_2 = -3.4476119$	$b_2 = 0.433679775$
$a_3 = -0.0922437$	$b_3 = -0.007217775$

Class I coefficients in the correction of $\dot{y}_{n+1}$

$a_1 = 1.474257443362816$	$b_0 = 0.0$
$a_2 = -1.082094997419516$	$b_\theta = 0.7329294937630792$
$a_3 = 0.3287359185634710$	$b_1 = 0.02556586336907029$
$a_4 = 0.2791016354932285$	$b_2 = 0.0$
	$b_3 = 0.5771268233940192$
	$b_4 = 0.07705956566094311$

Class II coefficients in the correction of y_{n+1}

$a_1 =$ 2.0580496972228	$b_0 =$ 0.0
$a_2 =$ - 0.963457924410107	$b_\theta =$ 0.1615959755323250
$a_3 =$ - 0.247233242848194	$b_1 =$ 0.704431783153855
$a_4 =$ 0.152641470035498	$b_2 =$ 0.0872587915934802
	$b_3 =$ - 0.154458906599228
	$b_4 =$ - 0.00951881093873159

Five-Step Algorithm (10^{th} Order) $\theta = .15$

Class I coefficients in the prediction of $\dot{y}_{n+1}$

$a_0 =$ - 79.1666666666667	$b_0 =$ 25.0
$a_1 =$ - 233.333333333333	$b_1 =$ 200.0
$a_2 =$ 100.0	$b_2 =$ 300.0
$a_3 =$ 191.666666666667	$b_3 =$ 100.0
$a_4 =$ 21.8333333333333	$b_4 =$ 5.0
$a_5 =$ 0.0	$b_5 =$ 0.0

Class II coefficients in the prediction of y_{n+1}

$a_0 =$ - 59.12658227848101	$b_0 =$ 4.898734177215190
$a_1 =$ - 121.3291139240506	$b_1 =$ 76.10126582278481
$a_2 =$ 362.9113924050633	$b_2 =$ 204.8354430379747
$a_3 =$ - 121.3291139240506	$b_3 =$ 76.10126582278481
$a_4 =$ - 59.12658227848101	$b_4 =$ 4.898734177215190
$a_5 =$ 1.0	$b_5 =$ 0.0

Class I coefficients in the prediction of $\dot{y}_{n+1-\theta}$

$a_0 = -$ 42.7695049029233	$b_0 =$ 14.30324426261697
$a_1 = -$ 118.4101058994344	$b_1 =$ 105.1481740386978
$a_2 =$ 53.88479833008057	$b_2 =$ 153.5716752407296
$a_3 =$ 97.33314699202555	$b_3 =$ 50.52574596664697
$a_4 =$ 10.96166548025159	$b_4 =$ 2.506754149118439
$a_5 =$ 0.0	$b_5 =$ 0.0

Class II coefficients in the prediction of $y_{n+1-\theta}$

$a_0 = -$ 26.73544108758085	$b_0 =$ 2.587417302238027
$a_1 = -$ 54.23037439023768	$b_1 =$ 35.31075452029697
$a_2 =$ 163.4154372805260	$b_2 =$ 93.55236974444488
$a_3 = -$ 52.98603813405493	$b_3 =$ 35.68803424310435
$a_4 = -$ 27.825532574613	$b_4 =$ 2.515509854023391
$a_5 = -$ 0.6380510940394066	$b_5 =$ 0.008547845603924952

Class I coefficients in the correction of $\dot{y}_{n+1}$

$a_1 = -$ 0.4295681469501183	$b_0 = -$ 0.08878758637340814
$a_2 =$ 0.6796954635322936	$b_\theta =$ 0.5693646365788814
$a_3 =$ 0.6817165393234353	$b_1 =$ 1.228232890045736
$a_4 =$ 0.06815614409438936	$b_2 =$ 1.187789665076307
$a_5 =$ 0.0	$b_3 =$ 0.3356259589243443
	$b_4 = -$ 0.01537141021047166
	$b_5 =$ 0.0

3. CYCLIC MULTISTEP CORRECTOR METHODS

3.1 INTRODUCTION

The cyclic methods recently developed by Donelson and Hansen [4,9] for Class I equations and those developed in this report for Class II equations compare very favorably with the traditional methods, indicating considerable potential for this approach.

Cyclic methods offer a distinct, new approach to multistep problems as in the case for off-grid and modified methods. Each of these new methods offers a modification of the classical approach which allows it to avoid the Dahlquist stability criterion [1] thus allowing the use of higher-order polynomials with fewer pack points. The modified methods correct back points as well as the current point. The off-grid methods evaluate the derivative at an additional off-grid point. A cyclic method uses different correctors applied in a cyclic manner to stabilize itself, allowing the individual correctors to be of higher order than would normally be possible for a classical stable method. An additional function evaluation is not required as for the off-grid methods. In fact, since fewer back points are used, potentially less computation is necessary than that required for traditional methods of the same order [e.g., Adams and Cowell methods].

In this report, the cyclic method as described in [4,9] for Class I equations has been extended to Class II equations. The Class II theory discussed is more complex than for Class I. For a given number of back points there exist

several methods each with different order and stability properties. Some strongly stable methods do not exist as opposed to the experience with Class I methods where no existence problems have occurred.

General procedures and computer programs are described for generating Class I and II cyclic correctors. These are used to thoroughly study existence, order, and stability of Class I and II methods with three and four back points.

For four back points a sixth-order Class II method has been derived. Computational results comparing this and another newly developed Class II method with Cowell's method for a two-body orbit problem are presented in Paragraph 3.6. This comparison shows the cyclic method to be far superior to Cowell's method of the same order. This is a significant result and when added to the above advantages indicates the value of further work on cyclic methods.

A summary of the analysis and comparisons performed to date is presented in paragraph 3.6. The tables referred to in this section are to be found in Appendix B.

3.2 THE CYCLIC METHOD

3.2.1 Class 1

Description

We wish to approximately solve

$$y'(x) = f(x,y),\ x \text{ in } [0,1],\ y(0) = y_0 \qquad (3\text{-}1)$$

where f satisfies a Lipshitz condition (e.g., the partial of f with respect to

y is bounded on the domain of f). We consider the unit interval only for the sake of convenient notation. The approximate solution at $x_n = nh$, $n = 1,2,\ldots,N$, will be denoted by y_n and $f(x_n,y_n)$ by y'_n. The step size is $h = 1/N$. We will cyclicly use M correcting methods of the form

$$\sum_{i=0}^{k} (a_i^m y_{n+i} - h\, b_i^m\, y'_{n+i}) = 0 \qquad (3\text{-}2)$$

where $m = 1,2,\ldots,M$. By the m^{th} corrector we will mean the m^{th} equation of the set (3-2). We assume $a_k^m \neq 0$. The number of back points used to correct y_{n+k} is k. Typically $k = 3,4,5,6$.

Given k starting values, $y_0, y_1,\ldots,y_{k-1}$, a predictor is used to estimate y_k. This y_k is corrected (perhaps several times) using the first corrector. The final corrected value is then used in the same predictor to estimate y_{k+1}. This y_{k+1} is corrected (again, perhaps several times) using the second corrector. This process is repeated until all M correctors are used. We next correct with the first corrector again and repeat the above cycle. Thus the name cyclic method. We will consider the case where $M = k$.

Order

"The m^{th} method is of order p_m" means the following $p_m + 1$ linear equations are satisfied:

$$C_i = 0 \qquad i = 0,1,\ldots,p_m \qquad (3\text{-}3)$$

where the formulae giving the C_i as linear combinations of the a_i^m and b_i^m are given in [8, p.221] where it is shown that a single method of order p has a

local discretization error of order h^{p+1} and a propagated error of order h^P.

Definition 1

The cyclic method (3-2) is of

(i) order P iff $P = \frac{1}{k} \sum_{m=1}^{k} P_m$

(ii) full order iff $P_m = 2k-1$, $m=1,\ldots,k$

(iii) optimal order iff $P = 2k$

The meaning of a fractional order, P, is answered in [9] for a special case. The general theoretical question is still open. Practical comparisons are given in a later section of this report. There was no problem in deriving stable Class I methods of full order; in fact, methods of order greater than full were derived. Notice that for $k \geq 3$, each method in a cyclic method of full order or greater is unstable by the Dahlquist theorem [1]; however, considered as a cyclic method we have k parameters remaining to satisfy a cyclic stability criteria as well as improve stability for nonzero h.

A cyclic method of optimal order for $k \geq 3$ will be unstable since it will consist of k identical, unstable, single methods, each of order 2k.

Stability

Let Y_s for $s = 0,1,\ldots,(N-k+1)/k$ be the k-dimensional vector of approximate solutions for each cycle (the number of the cycle being s and the starting values being Y_0) and U and L, the kxk matrices that follow:

$$Y_s = (y_{sk}, y_{sk+1}, \ldots, y_{sk+k-1})^T$$

$$U = \begin{bmatrix} a_0^1 & a_1^1 & a_2^1 & \cdots & a_{k-1}^1 \\ 0 & a_0^2 & a_1^2 & \cdots & a_{k-2}^2 \\ 0 & 0 & a_0^3 & \cdots & a_{k-3}^3 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & a_0^k \end{bmatrix}$$

$$L = \begin{bmatrix} a_k^1 & 0 & 0 & \cdots & 0 \\ a_{k-1}^2 & a_k^2 & 0 & \cdots & 0 \\ a_{k-2}^3 & a_{k-1}^3 & a_k^3 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ a_1^k & a_2^k & a_3^k & \cdots & a_k^k \end{bmatrix} \qquad (3\text{-}4)$$

Then for h = 0 (3.2) becomes [4]

$$Y_{s+1} = A\,Y_s = A^{s+1}Y_0 \qquad (3\text{-}5)$$

where $A = -L^{-1}U$. Let $p(\lambda)$ be the characteristic polynomial of A, the stability polynomial. Then $p(\lambda)$ plays the role of the "α" polynomial, $\rho(\zeta)$ in [8 , p.218] and we have the corresponding definition.

Definition 2

The cyclic method (3-2) is:

(i) stable iff. the modulus of each root of $p(\lambda)$ is ≤ 1 and roots of modulus 1 are simple

(ii) strongly stable iff. it is stable and there is only one root of modulus 1. We call this root the principal root, the other roots are extraneous.

(iii) weakly stable iff. all roots are on the unit circle and it is stable.

The stability condition $p(\lambda) = 0$ can be written in the form

$$\det (\lambda L+U) = 0 \qquad (3\text{-}6)$$

by premultiplying $A-\lambda I$ by L, making use of properties of the determinant, and using $a_k^m \neq 0$ for $m = 1,\ldots,k$ so that $\det L \neq 0$. In this form, it is easy to see that the constant term $= a_0^1 \ldots a_0^k$ so zero is a root iff one $a_0^m = 0$, the coefficient of $\lambda^k = a_k^1 \ldots a_k^k \neq 0$ so the polynomial is always of order k, and if each method is at least of order one then 1 is at least a simple root. The latter can be seen by noticing that the rows of $\lambda L+U$ sum to zero.

If we consider the equation $y' = 0$, $y(0) = 1$, we know from Faddeev that it is necessary that the roots of $p(\lambda)$ be ≤ 1 and at least one equal to 1 in modules for convergence. Also from experience with equations of the form (3-5) and some experience with cyclic methods with large extraneous roots [9] we know the extraneous roots must be small in modulus or they can cause the propagated effects of starting and local errors to ruin the approximate solution. Intuitively, the smaller the roots, the more strongly stable the method. Methods with extraneous roots, all zero, will be called most strongly stable. It is therefore surprising that our early experience with cyclic methods have indicated a weakly stable cyclic method to be better than a strongly stable

one of slightly lower order (see paragrpgh 3.6).

Convergence

If each m^{th} method satisfies $h < \left|\frac{a_k^m}{b_k^m \mathcal{L}}\right|$ then the corrector iteration at a fixed x_n will yield the unique solution of each corrector difference equation to arbitrary accuracy in a convergent process [8].

To justify that the fully corrected approximate solution converges to the exact solution of the differential equation as h approches zero, we paraphrase the theorem of Donelson and Hansen [9].

Theorem 1

If (i) the differential equation satisfies a Lipschitz condition with constant $\mathcal{L}$

(ii) its exact solution has 2k continuous derivatives

(iii) the cyclic method is of full order

(iv) it is stable

(v) the local discretization error in the starting values is of order h^{2k}

(vi) h is "small enough"

then the propagated error at each x_n is of order h^{2k-1} and the error bound increases with x_n.

3.2.2 Class II

We wish to approximately solve

$$y''(x) = f(x,y), x \in [0,1], y(0) = y_0, y'(0) = y'_0 \qquad (3\text{-}7)$$

where f satisfies a Lipschitz condition in y. Using the same notation as for Class I, we will use k correcting methods of the form

$$\sum_{i=0}^{k} (a_i^m y_{n+i} - h^2 b_i^m y''_{n+i}) = 0 \ . \qquad (3\text{-}8)$$

This cyclic method is used as in Class I. We assume $a_k^m \neq 0$.

Order

"The m^{th} method is of order p_m" means the following $p_m + 2$ linear equations are satisfied

$$C_i = 0 \qquad i = 0,1,\ldots,p_m+1 \qquad (3\text{-}9)$$

where the formulae giving the C_i as linear combinations of the a_i^m and b_i^m are given below (the formulae in [8 ,p.296] are in error).

$$C_0 = a_0 + a_1 + \ldots + a_k$$
$$C_1 = a_1 + 2a_2 + \ldots + ka_k$$
$$2!C_2 = a_1 + 2^2 a_2 + \ldots + k^2 a_k - 1.2(b_0 + b_1 + \ldots + b_k)$$
$$3!C_3 = a_1 + 2^3 a_2 + \ldots + k^3 a_k - 2.3(\ b_1 + \ldots + kb_k)$$
$$(p+1)!C_{p+1} = a_1 + 2^{p+1} a_2 + \ldots + k^{p+1} a_k - p(p+1)(b_1 + \ldots + k^{p-1} b_k)$$

Henrici shows that a single method of order p has a local discretization error of order h^{p+2} and a propagated error of order h^p.

Definition 3

The cyclic method (3-8) is of

(i) order P iff. $P = \frac{1}{k} \sum_{m=1}^{k} P_m$

(ii) full order iff. $P_m = 2k-2$, $m = 1,\ldots,k$

(iii) optimal order iff. P = 2k-1

The meaning of fractional order is still an open question. Practical comparisons are given in a later section of this report. For example, a cyclic method of order 5 3/4 for k = 4 is much better than Cowell's method of order 5. There were problems encountered in attempting to derive strongly stable Class II methods of full order for $k \geq 4$; however, some very suitable methods were derived. For $k \geq 3$, a single method of order 2k-1 will be unstable by the Dahlquist theorem; however, in a cyclic method with one or more such single methods we have several parameters remaining to satisfy a cyclic stability criteria as well as to improve stability for nonzero h. A cyclic method of optimal order for $k \geq 3$ will be unstable since it will consist of k identical, unstable, single methods each of order 2k-1.

Stability

Using the same notation as for Class I, we derive (3-5) again for h = 0 and the same stability polynomial plays the role of the "α" polynomial of [8 ,p.300].

Definition 4

The cyclic method of (3-8) is

(i) stable iff. the modulus of each root of $p(\lambda)$ is ≤ 1 and roots of modulus 1 have multiplicity of at most two,

(ii) strongly stable iff. it is stable and there is only one root of modulus 1 with multiplicity two. The root is called the principal root, the others are extraneous.

There may be simple extraneous roots of modulus 1 in a strongly stable Class II method. The derivation of, and all the conclusions following (3-6) for Class I carry over to Class II with no changes except that we have found that 1 is a root of multiplicity two as expected.

Convergence

If each m^{th} method satisfies

$$h^2 < \left| \frac{a_k^m}{b_k^m \mathcal{L}} \right|$$

then the corrector iteration at each x_n will yield the unique solution of each corrector difference equation to arbitrary accuracy in a convergent process [8 ,p.299].

We believe the Class II convengence theorem of [8, p.314] can be extended to cyclic methods as follows:

Theorem 2

If (i) the differential equation satisfies a Lipschitz condition with constant $\mathcal{L}$

(ii) its exact solution has 2k continuous derivatives

(iii) the cyclic method is of full order (Class II)

(iv) it is stable (Class II)

(v) the local discretization error in the starting values is of order h^{2k}

(vi) h is "small enough" (not necessarily $\to 0$)

then the propagated error at each x_n is of order h^{2k-2} and the error bound increases with x_n.

3.3 COMPUTATION OF THE COEFFICIENTS

3.3.1 Class I

Normalization

In the m^{th} corrector, we have k+1 a's and k+1 b's, leaving 2k(k+1) coefficients to determine. Since in any one method we can multiply (3-2) by any nonzero constant leaving the method essentially unchanged and since we must divide by a_k^m each time we correct, we will take as our k normalization conditions

$$a_k^m = 1, m=1,\ldots,k \qquad (3\text{-}10)$$

leaving k(2k+1) degrees of freedom.

Order

We will begin by attempting to compute coefficients for cyclic methods of full order. For each method we have 2k order equations to satisfy (3-3) taken together we have $2k^2$ linear order equations leaving k degrees of freedom. We found it convenient to solve the order equations for each method in terms of the two parameters a_k^m and a_0^m. The first is used for normalization and the second to satisfy stability and perhaps additional order or other conditions. These parametric solutions are given in Appendix B, table B.1.

Stability

To minimize the propagation of starting and local errors, we will force the extraneous roots to be zero. It seems necessary to either write L^{-1} in terms of the symbols a_i^m, multiply symbolically by U and expand $\det(L^{-1}U+\lambda I)$ symbolically in powers of λ or expand (3-6) symbolically in powers of λ. The latter course seems the easiest. This was done by hand for k=3, 4, and 5 and is given in table B.5. However, for larger k it may pay to use a symbol manipulation computer language such as FORMAC [10].

For each k we must derive expressions for the following F's in terms of the symbols a_i^m,

$$p(\lambda) = F_k \lambda^k + F_{k-1} \lambda^{k-1} + \ldots + F_0 = 0 \qquad (3\text{-}11)$$

We then have our k-1 nonlinear stability equations:

$$F_0 = F_1 = \ldots = F_{k-2} = 0 \qquad (3\text{-}12)$$

leaving 1 degree of freedom which may be used in several ways. For k=4, it was used to satisfy one more order equation. Since $F_0 = a_0^1 \cdot a_0^2 \ldots a_0^k$ choosing one $a_0^m = 0$ will satisfy the first stability equation yielding one zero root immediately and leaving k-2 stability equations.

Algorithm 1

We select the 2k(k+1) parameters a_i^m, b_i^m, m=1,2,...,k; i=0,1,...,k, of the cyclic method (3-2) for a given k by the following automated procedures

(i) Solve the 2k linear order equations in terms of the two parameters a_k^m and a_0^m for each method using a computer program

(ii) Satisfy the k normalization conditions

(iii) Choose one a_o^m to satisfy another condition

(iv) Choose one $a_o^m = 0$

(v) Substitute (i) - (iv) into a computer program which solves the k-2 stability equations for the k-2 unknowns using an iterative method.

(vi) Substitute these solutions into (i) to obtain the full set of coefficients.

These programs are described in Appendix B.

3.3.2 Class II

Normalization and Order

We again assume the same normalization conditions leaving k(2k+1) degrees of freedom. We will begin by attempting to compute coefficients for Class II cyclic methods of full order. We have $2k^2$ linear order equations (3-9), leaving k degrees of freedom. We found it convenient to solve the order equations for each method in terms of the two parameters a_k^m and a_0^m. For k=4 with full order it was necessary to leave a_4^m and a_2^m as two parameters since symmetry in the solution of the order equations dictates $a_0^m = a_4^m$ and it is not possible to let them both vary as "free" parameters (the order equation matrix in this case did not invert). These parametric solutions are given in table B.2 in Appendix B.

Stability

To minimize the propagation of starting and local errors, we will attempt to satisfy the strongest stability condition that two of the roots of $p(\lambda)$ are 1

and the others are 0. Since $p(\lambda)$ is the same as for Class I, we can use the same algebraic expansions for the F_i (3-11). We then have k-2 nonlinear stability equations:

$$F_0 = F_1 = \ldots = F_{k-3} = 0 \qquad (3\text{-}13)$$

leaving two degrees of freedom which may be used in several ways. For k=3, they were used to satisfy two more order equations. Choosing one $a_0^m = 0$ will satisfy the first stability equation yielding one zero root.

Algorithm 2

When possible, we select the $2k(k+1)$ parameters a_i^m, b_i^m, m=1, ..., k; i=0,...,k, of the cyclic method (3-8) for a given k by

(i) Solving the 2k linear order equations in terms of the two parameters a_k^m and a_0^m for each method using a computer program

(ii) Satisfying the k normalization conditions

(iii) Choosing a_0^2 and a_0^3, for example, to satisfy other conditions;

(iv) Choose one $a_0^m = 0$; e.g., a_0^1

(v) Substitute (i) - (iv) into the remaining k-3 nonlinear stability equations and solve k-3 equations in k-3 unknowns using an iterative method.

Computer programs for performing this procedure are described in Appendix B.

3.4 RESULTS OF THE COMPUTATION OF THE COEFFICIENTS

3.4.1 Three Back Points

Stability Polynomial

Writing (3-6) for k=3 and expanding, we find

$$p(\lambda) = \begin{vmatrix} \lambda a_3^1 + a_0^1 & a_1^1 & a_2^1 \\ \lambda a_2^2 & \lambda a_3^2 + a_0^2 & a_1^2 \\ \lambda a_1^3 & \lambda a_2^3 & \lambda a_3^3 + a_0^3 \end{vmatrix} \qquad (3\text{-}14)$$

$$= F_3\lambda^3 + F_2\lambda^2 + F_1\lambda + F_0 \qquad \text{where}$$

$$F_3 = a_3^1\, a_3^2\, a_3^3$$

$$F_2 = a_0^1\, a_3^2\, a_3^3 - a_1^1\, a_2^2\, a_3^3 + a_2^1\, a_2^2\, a_2^3 - a_2^1\, a_3^2\, a_1^3 + a_3^1\, a_0^2\, a_3^3 - a_3^1\, a_1^2\, a_2^3 + a_3^1\, a_3^2\, a_0^3$$

$$F_1 = a_0^1\, a_0^2\, a_3^3 - a_0^1\, a_1^2\, a_2^3 + a_0^1\, a_3^2\, a_0^3 + a_1^1\, a_1^2\, a_1^3 - a_1^1\, a_2^2\, a_0^3 - a_2^1\, a_0^2\, a_1^3 + a_3^1\, a_0^2\, a_0^3$$

$$F_0 = a_0^1\, a_0^2\, a_0^3$$

Class I Coefficients

Applying algorithm 1 with $a_0^1 = a_0^3 = 0$ the equation $F_1(a_0^2) = 0$ was solved for a_0^2. In this case the nonlinear equation reduced to a linear equation so we of course arrived at exactly the same solution

$$a_0^2 = +\ 1.504166\ldots = 3249/2160 \qquad (3\text{-}15)$$

for all six positive and negative starting values.

These numbers agree with those of [4,9]. This provides a check on our automated procedure. Each method is of order 5 so the cyclic method is of order 5. These coefficients were substituted into $p(\lambda)$ (3-14), yielding

$$p(\lambda) = \lambda^3 - \lambda^2 = 0 \tag{3-16}$$

which has roots 1,0,0 as desired. Since each individual method exceeds the maximum allowable order for a stable single method, each would be unstable if used alone. In fact the characteristic polynomials for the single methods are:

$$\begin{aligned} &\lambda^3 + (0.727272...)\,\lambda^2 - (1.727272...)\,\lambda + 0 && ,m=1,3 \\ &\lambda^3 - (1.870833...)\,\lambda^2 - (0.633333...)\,\lambda + (1.504166...) && ,m=2 \end{aligned} \tag{3-17}$$

with roots - 1.727272...,1,0 for m=1,3 and 1.741666...,1,-0.870833... for m=2; clearly unstable. This most strongly stable cyclic method is composed of three unstable single methods.

Comparison was made in [4] with the off-grid method of the same order by numerically solving the same differential equation using the two methods and by computing values of the known analytic solution. The accuracies are comparable but the cyclic method computing time is shorter since one less function evaluation is necessary.

The choice $a_0^3 = 0$ was arbitrary and this parameter can now be used, for example, to increase order. This was done in [9] to derive a cyclic method composed of single methods of orders 6,5,5 but the roots of $p(\lambda)$ of this cyclic method are all on the unit circle yielding a weakly stable method. Nevertheless, this second method does yield smaller errors. Using the automated procedure it would be no problem to derive a cyclic method of orders 6, 5, 5 with zero extraneous roots.

Class II Coefficients

Applying Algorithm 2 with $a_0^m = 0$, m = 1,2,3, we satisfy $F_0 = 0$. There were no remaining nonlinear equations. The resulting coefficients for the three methods are exactly those for the Cowell traditional method so the cyclic method is exactly Cowell's which is of order 4 with roots 1,1,0. The cyclic method is therefore order 4 and upon substituting the coefficients into $p(\lambda)$ (3-14) we find roots 1,1,0 for the cyclic method of full order also.

The choice $a_0^1 = a_0^2 = 0$ was arbitrary and we can now use these parameters to increase order.

Again applying Algorithm 1, using table B.1-1 with $a_0^3 = 0$ and table B.1-2 for m=1 and 2 we satisfy $F_o = 0$. There were no remaining nonlinear equations. The resulting coefficients are given in table B.4-1.

These coefficients are new so there is no direct comparison with the results of other authors. The first and second methods are of order 5, the third, 4, so the cyclic method is of order 4 2/3, greater than full order. Note that this method is "nearly" optimal. These coefficients were substituted in $p(\lambda)$ (3-14) yielding

$$p(\lambda) = \lambda^3 - 2\lambda^2 + \lambda = 0 \qquad (3\text{-}18)$$

which has roots 1,1,0 as desired. Since the first two individual methods exceed the maximum allowable order for a stable single method, each would be unstable if used alone. In fact the characteristic polynomials for the single methods are:

$$\lambda^3 - 3\lambda^2 + 3\lambda - 1 = 0, m = 1,2 \text{ with roots } 1,1,1; \qquad (3\text{-}19)$$

unstable (not convergent) by definition since multiplicity is greater than two. This most strongly stable cyclic method is composed of one stable and two unstable single methods.

Comparisons were made with Cowell's method for k = 3 by numerically solving a two body orbit problem. The cyclic method gave more accurate results for the same number of starting values, the same step size, and the same computing time.

3.4.2 Four Back Points

Stability Polynomial

Writing (3-6) for k=4 and expanding, we find:

$$p(\lambda) = \begin{vmatrix} \lambda a_4^1 + a_0^1 & a_1^1 & a_2^1 & a_3^1 \\ \lambda a_3^2 & \lambda a_4^2 + a_0^2 & a_1^2 & a_2^2 \\ \lambda a_2^3 & \lambda a_3^3 & \lambda a_4^3 + a_0^3 & a_1^3 \\ \lambda a_1^4 & \lambda a_2^4 & \lambda a_3^4 & \lambda a_4^4 + a_0^4 \end{vmatrix} \qquad (3\text{-}20)$$

$$= F_4\lambda^4 + F_3\lambda^3 + F_2\lambda^2 + F_1\lambda + F_0$$

where the formulae for the F_I and the above determinant are given in table B.5-1 on computer output in FORTRAN notation. Numerical checks were made of the algebraic expansions to insure their validity.

Class I Coefficients

Applying Algorithm 1, with $a_0^1 = a_0^3 = 0$, the equations $F_1(a_0^2, a_0^4) = 0$ and $F_2(a_0^2, a_0^4) = 0$ were solved simultaneously for a_0^2 and a_0^4. For all eight sets of positive and negative starting values varying in magnitude from 1 to 144 we arrived at the same solution to 25 places

$$a_0^2 = +2.813246... \qquad (3\text{-}21)$$

$$a_0^4 = +0.842603...$$

with the interesting result that the solutions may be interchanged, i.e., $a_0^4 = 2.813...$, $a_0^2 = 0.842..$ is also a solution. This means the second and fourth methods may be interchanged, a reasonable result when one considers the definition of the cyclic method. The coefficients are given in table B.3-2.

These numbers agree with those of [4] (modlo a typographical error in the sign of a_4^4 on p.138) which were derived in a different manner. These facts imply that the coefficients are unique up to a permutation of the methods. Each method is of order 7, so the cyclic method is of order 7. These coefficients were substituted in $p(\lambda)$ yielding

$$p(\lambda) = \lambda^4 - \lambda^3 = 0 \qquad (3\text{-}22)$$

which has roots 1,0,0,0. Since each single method exceeds the maximum allowable order for a stable single method, each would be unstable if used alone. In fact the characteristic polynomials for the single methods are:

$$\lambda^4+(3.927272...)\lambda^3-(2.454545...)\lambda^2-(2.472727...)\lambda+0, m=1,3$$
$$\lambda^4-(3.029119...)\lambda^3-(9.359787...)\lambda^2+(8.575660...)\lambda+(2.813246...), m=2$$
$$\lambda^4+(1.843742...)\lambda^3-(4.522755...)\lambda^2+(8.364083...)\lambda+(0.842603...), m=4 \qquad (3\text{-}23)$$

with roots - 4.36...,1, - 0.57...,0 for m=1,3;

- 3.28...,1,0.77..., - 0.33... for m=4; and 4.62...,

- 2.33...,1, - 0.26... for m=2 ; all unstable

therefore not convergent. This most strongly stable cyclic method is composed of four unstable single methods.

Comparison was made with the off-grid method of the same order [4]. The accuracies are comparable but the cyclic method computing time is shorter since one less function evaluation is necessary.

The choice $a_0^1 = 0$ was arbitrary and this parameter can now be used to increase order. A method of order 7 1/4 was derived in [9] at the expense of creating some sizeable, nonzero extraneous roots. Applying algorithm 1 using table B.1-3 for higher order for m = 1 we obtain

$$a_0^2 = +2.459251...$$
$$a_0^4 = +0.634992... \qquad (3\text{-}24)$$

again with the result that the solutions may be interchanged. This result implies the coefficients are unique up to a permutation of methods. The coefficients are given in table B.3-3.

The single methods are of order 8,7,7,7 so the cyclic order is 7 1/4. These coefficients were substituted in $p(\lambda)$ yielding $p(\lambda) = \lambda^4 - \lambda^3 = 0$ which has roots 1,0,0,0.

Since each single method exceeds the maximum allowable order for a stable single method, each would be unstable if used alone. In fact the characteristic polynomials for the single methods are:

$$\begin{aligned}
&\lambda^4 + 6.4\lambda^3 + 0\lambda^2 - 6.4\lambda - 1 && m=1\\
&\lambda^4-(2.153784\ldots)\lambda^3-(8.490889\ldots)\lambda^2+(7.185422\ldots)\lambda+(2.459251\ldots) && m=2\\
&\lambda^4+(3.927272\ldots)\lambda^3-(2.454545\ldots)\lambda^2-(2.472727\ldots)\lambda+0 && m=3\\
&\lambda^4+(2.357108\ldots)\lambda^3-(4.013164\ldots)\lambda^2+(0.021062)\lambda+0.634992 && m=4
\end{aligned} \qquad (3\text{-}25)$$

with roots $-6.23\ldots,1,-1,-.17\ldots$ for m=1; approx. $3.9,-2.3,1,-.4$ for m=2; $-4.36\ldots,1,-0.57\ldots,0$ for m=3; and approx. $-3.5,1,.5,-.3a$ for m=4; all unstable therefore not convergent. This most strongly stable cyclic method is composed of four unstable single methods.

Class II Existence

From table B.2-4 we see that $a_0^m = a_4^m$, m=1,...,4 due to symmetry in the solution of the Class II order equations for k=4, order 6. Since $F_0 = a_0^1\ a_0^2\ a_0^3\ a_0^4$, no Class II method of full order for k=4 with a zero extraneous root exists. Does any strongly stable method of full order exist? To answer this question we must study the properties of the roots of $p(\lambda)$ when table B.2-4 is used.

So far for k=3 and 4, we have completely avoided the Dahlquist stability condition; i.e., strongly stable cyclic methods of order higher than that possible for even weakly stable single methods were derived. We now will establish a result for k=4 similar to part of the stability theorem 6.5 of [8, p.307].

Theorem 3

(i) A necessary condition for the existence of a stable cyclic method of full order or better for k=4 is that all roots of $p(\lambda)$ have modulus 1.

(ii) Such a method does exist and is composed of optimal single methods; i.e., methods of full order or better which are at best weakly stable.

Proof

(i) By "full order or better" we mean the order of each single method is ≥ 6. Since $p(\lambda)$ is necessarily of order 4 with a double root of 1, it must be of the form

$$\begin{aligned} p(\lambda) &= (\lambda^2-2\lambda+1)(\lambda-a)(\lambda-b) \\ &= \lambda^4+(-2-a-b)\lambda^3+(1+2a+2b+ab)\lambda^2 \\ &\quad +(-a-b-2ab)\lambda+ab \end{aligned} \tag{3-26}$$

where a and b are the two arbitrary roots.

The coefficients of the stability polynomial were mapped out as indicated in B.5-1. The surprising result was that in all cases $F_0 = F_4 = 1$, $F_1 = F_3$, and $F_2 = 2F_1-2$ to 25 places. This is the same sort of symmetry about the middle value as that exhibited by the optimal single methods of [8, p.311]. We concluded that the symmetry in the order equations caused the symmetry in the coefficients of the stability polynomial. No matter how we choose the free parameters we cannot avoid this symmetry. Only one degree of freedom can be applied to vary the F's, whereas we had expected to use two.

Since $F_0 = 1$, $ab = 1$, and $b = 1/a$, if either a or b is within the unit circle, the other is outside and the method will not be stable at all. All roots of a stable method must be on the unit circle. Notice (3-26) now becomes $p(\lambda) = \lambda^4 - (2a+1/a)\lambda^3+(2+2a+1/a)\lambda^2 - (2+a+1/a)\lambda + 1$ which has symmetric coefficients. Also note that this proof works for a single method as well as for cyclic methods for k=4.

(ii) will be established by deriving the coefficients.

Class II Coefficients

Applying algorithm 2 with $a^1_2 = a^3_2 = -2$, and $a^4_2 = 0$ we get

$$p(\lambda) = \lambda^4 - 2\lambda^2 + 1 = 0 \qquad (3\text{-}27)$$

which has roots 1,1,-1,-1. a^2_2 is still a free parameter. It was chosen =0 arbitrarily but could have been used to increase one of the methods to order 7. The results are given in table B.4-2. Since each single method is of maximal order for a stable single method, each would be weakly stable if used alone. In fact the characteristic polynomials for the single methods are:

$$1\lambda^4 + 0\lambda^3 - 2\lambda^2 + 0\lambda + 1 \ , m=1,3$$
$$1\lambda^4 - 1\lambda^3 + 0\lambda^2 - 1\lambda + 1 \ , m=2,4 \qquad (3\text{-}28)$$

with roots 1,1,-1,-1 for m=1,3 and $1,1,-\frac{1}{2}\pm\frac{1}{2}\sqrt{3}i$ for m=2,4

These optimal methods are given in [8 ,p.311]. This weakly stable cyclic method is composed of four weakly stable single methods. This coincidence is due to the fact that for k=4, 2k-2 = k+2. This would not occur for k=5. Comparisons were made with Cowell's method for k=4 by numerically solving a two-body orbit problem. The cyclic method presented greater accuracy for the same step size. These results are discussed in detail in paragraph 3.6.

As mentioned above we suspect only one degree of freedom is necessary to "solve" the stability equations (e.g., force $F_2 = 0$) leaving three degrees of freedom and the possibility of deriving a weakly stable cyclic method of order 6 3/4. We know it is possible to get one of order 6 1/4 as was shown in the above proof.

Another option was explored to find a strongly stable method for k=4; i.e., decrease the order of one of the methods, let $a_0 = 0$ for this method, solve the other stability equations, and increase order with any remaining degrees of freedom. Applying Algorithm 2, with $a_2^2 = 1$, $a_0^2 = a_2^1 = a_2^3 = a_0^4 = 0$ in table B.2-3 for m=2 and B.2-4 for m=1,3,4, yields

$$p(\lambda) = \lambda^4 - 2\lambda^3 + \lambda^2 = 0 \qquad (3\text{-}29)$$

which has roots 1,1,0,0. The coefficients are given in table B.4-3. The single methods are of orders 6,5,6,6 so the cyclic order is 5 3/4. Method 2 is a Cowell method and methods 1,3, and 4 are Henrici's optimal weakly stable method with roots all on the unit circle. This most strongly stable cyclic method is composed of one most strongly stable and three weakly stable single methods.

Since we have four parameters and need to satisfy only two remaining nonlinear equations, it may be possible to increase the orders of two methods to obtain a most strongly stable cyclic method composed of single methods of orders 7,7,6,5 for a cyclic order of 6 1/4 thus, in a certain sense, again avoiding the stability theorem. Future work will consider this option.

3.4.3 Five Back Points

Stability Polynomial

The determinant (3-6) for k=5 was expanded as for k=4. The formulae for the F's and the determinant are given in table B.5-2 on computer output in FORTRAN notation. The algebra was checked numerically as for k=4 and the relative error never exceeded 10^{-25} indicating that the expansion is correct.

Class II Existence

From table B.2-5 we see that it is not necessary to choose $a_0 = a_5 = 1$ as it was for k=4. To derive a method of full order with zero extraneous roots we must choose one $a_0^m = 0$. To study the properties of the roots of $p(\lambda)$ for a

method of full order, we chose several sets of arbitrary values of the a_0^m with at least one $a_0 = 0$ in each set, calculated the other coefficients using table B.2-5 with $a_5^m = 1$, and calculated the F values from B.5-2. The result was that in all cases

$$F_0=0, F_1=F_5=1, F_2=F_4=7526.331\ldots, F_3=-2F_1-2F_2=-15054.663\ldots .$$

We again conclude that the symmetry in the solution of the order equations caused the symmetry in $p(\lambda)$. We notice that the roots of this polynomial are approximately -7528,1,1.1,0. Given that one $a_0^m = 0$, no matter how we choose the remaining parameters we cannot get all roots within the unit circle. We conclude the following:

Lemma

No class II method of full order for k=5 with a zero extraneous root exists.

This is the same conclusion we first reached for k=4 for slightly different reasons. We suggest that, as for k=4, it is possible to derive a weakly stable method of full order and a most strongly stable method of almost full order. These derivations will not entail much additional labor since the expansion for $p(\lambda)$ has already been obtained.

3.5 ANALYSIS FOR NONZERO STEP SIZE

The stability criterion used in the previous sections was developed independent of the step size h. In practical applications, however, the step size is very important. For a specified level of accuracy, the method using the largest

step size will require the least computation. It is this important to study the effects of finite values of h on the stability properties of the developed methods with the hope that this analysis can lead to further improvements.

To begin the analysis for $h \neq 0$ write the k Class I methods (3-2) in matrix form,

$$LY_{s+1} + UY_s - h[L'Y'_{s+1} + U'Y'_s] = 0 \qquad (3\text{-}30)$$

where Y is defined in (3-4), Y' is composed of the y'_n's, U and L are defined by (3-4) and U' and L' are the same as U and L except that the a_k^m are replaced by the b_k^m.

The first use of this matrix form as a one-step corrector difference equation is to derive a cyclic analogue of the corrector iteration convergence theorem [8].

Since L is nonsingular, we have $Z = F(Z)$ from (3-30) where $Z = Y_{s+1}$. Fixed point iteration will converge if F is a contracting map. Since F is Lipschitz in y with constant $\mathcal{L}$ we have

$$||F(Z^1)-F(Z^2)|| \leq h||L^{-1}L'||\ ||(f(y^1_{sk+k})-$$

$$f(y^2_{sk+k}),\ldots,f(y^1_{s+2k-1})-f(y^2_{s+2k-1}))\ ||$$

$\leq h||L^{-1}L'||\alpha||Z^1-Z^2||$. So the corrector iteration will converge if $h<1/\alpha||L^{-1}L'||$. The condition for a single method is $h<1/\alpha|b_k\ a_k^{-1}|$.

The second use of this form of the difference equation is to study stability for $h \neq 0$ by applying an analysis similar to that of [12] for a single method to each component of (3-30) A preliminary analysis indicates that the stability condition for $h \neq 0$ is very similar to that for $h = 0$; so similar in fact that it seems no new algebraic expansion such as that of table B.5 is needed, thus deriving stable methods for fixed $h \neq 0$ will be only slightly more complicated than for $h = 0$.

Another use of (3-30) is to solve it or bound the solution thus obtaining an expression or an upper bound to the propagated error. It seems that this method of finding error bounds may be easily extended to Class II. A

preliminary look indicates that fractional order just means the average order over all methods if the L^1 norm of the error is used.

3.6 SUMMARY AND DISCUSSION OF RESULTS

The basic theory of Class I cyclic methods is described in paragraph 3.2.1. The analysis presented parallels that of Donelson and Hansen [9]. In paragraph 3.2.2 this theory is extended to Class II methods. Algorithms for computing Class I and II cyclic method coefficients are given in paragraph 3.3. Using these algorithms and the computer programs developed for this study, coefficients for Class I and II cyclic methods with three and four back points are derived in paragraphs 3.4.1 and 3.4.2. In paragraph 3.4.3, study is initiated of Class II methods with five back points. The stability polynomial is derived and the non-existence of a Class II five-point method of full order with an extraneous zero root is demonstrated. Some preliminary results for nonzero step size are presented in paragraph 3.5. Computer programs used to generate coefficients for Class I and II methods are described in Appendix B. Also in Appendix B are tables providing parametric solutions to the order equations, coefficients of the newly developed cyclic methods, and stability polynomials for the four and five step cases.

Table 2 summarizes the cyclic methods developed for three and four back points. Methods 1 and 2 are identical with those presented by Donelson and Hansen in [9]. Method 3 is new. Methods 1 and 2 have been compared by Donelson and Hansen to Class I off-grid methods of equal order. These authors indicate that

method 2 was usually better in most cases while method 1 was usually worse than its off-grid counterpart. We have done some numerical comparisons for methods 5, 6 and 7. Preliminary results are very encouraging. Results of the comparisons of methods 6 and 7 to a Cowell method of equal order are presented in Table 3, and plotted in figure 19.

As is indicated in Table 3, the cyclic method 7 is superior to its Cowell counterpart for all step sizes. This is true even when the Cowell method is used with a higher-order predictor than that used with the cyclic method (e.g., Störmer order 6 with the Cowell method versus Störmer order 5 with the cyclic method). This is significant since only one corrector iteration was made for each method. Cyclic method 6 is also superior to its counterpart the Cowell 5th-order method but is considerably inferior to the 6th-order Cowell method and cyclic method 7. The relation for these methods between error and step size is plotted in figure 19.

Table 2. Summary of Cyclic Methods

Method Number	Number of Back Points k	Cyclic Method Order P	Order of Individual Methods	Roots of the Cyclic Stability Polynomial
		---CLASS I---		
1	3	5	5,5,5	1,0,0
2	4	7	7,7,7,7	1,0,0,0
3	4	7 1/4	8,7,7,7	1,0,0,0
		---Class II---		
4	3	4	4,4,4	1,1,0
5	3	4 2/3	5,5,4	1,1,0
6	4	5 3/4	6,5,6,6	1,1,0,0
7	4	6	6,6,6,6	1,1,-1,-1
8	4*	6*1/4	7,7,6,5*	1,1,0,0*

* The coefficients for this method were not derived, although the theory indicates the method might be feasible.

Table 3. Comparison of Cyclic and Cowell Methods
GEOS B Satellite Two-Body Motion

Order of Predictor / Order of Corrector		Störmer-5 Cowell-5	Störmer-5 Cowell-6	Störmer-6 Cowell-6	Störmer-5 Cyclic-5 3/4	Störmer-5 Cyclic-6
Step Size (Secs.)	Length of Arc (Hours)	PROPAGATED ERROR (METERS)*				
50	2	0.63	0.022	0.018	0.15	0.003
	12	11.30	0.18	0.12	2.68	0.02
	24	57.5	0.47	0.30	8.95	0.06
	36	73.2	0.82	0.49	17.5	0.11
	48	112.2	1.18	0.67	26.9	0.15
100	2	20.5	1.65	1.37	3.98	0.21
	12	366.7	16.9	11.7	76.6	1.98
	24	1,216.0	47.5	30.0	257.6	5.45
	36	2,369.8	86.4	52.0	504.8	9.90
	48	3,637.7	127.5	74.5	776.7	14.37
150	2	159.6	21.0	18.7	24.9	2.20
	12	2,825.9	234.5	180.0	491.2	22.3
	24	9,361.0	682.1	491.3	1,661.0	62.4
	36	17,261.6	1,259.4	880.0	3,261.1	113.2
	48	28,105.3	1,877.0	1,287.8	5,025.4	166.9

* Propagated errors are taken to be the absolute value of the difference between the analytic solution and the approximated solution for the X coordinate.

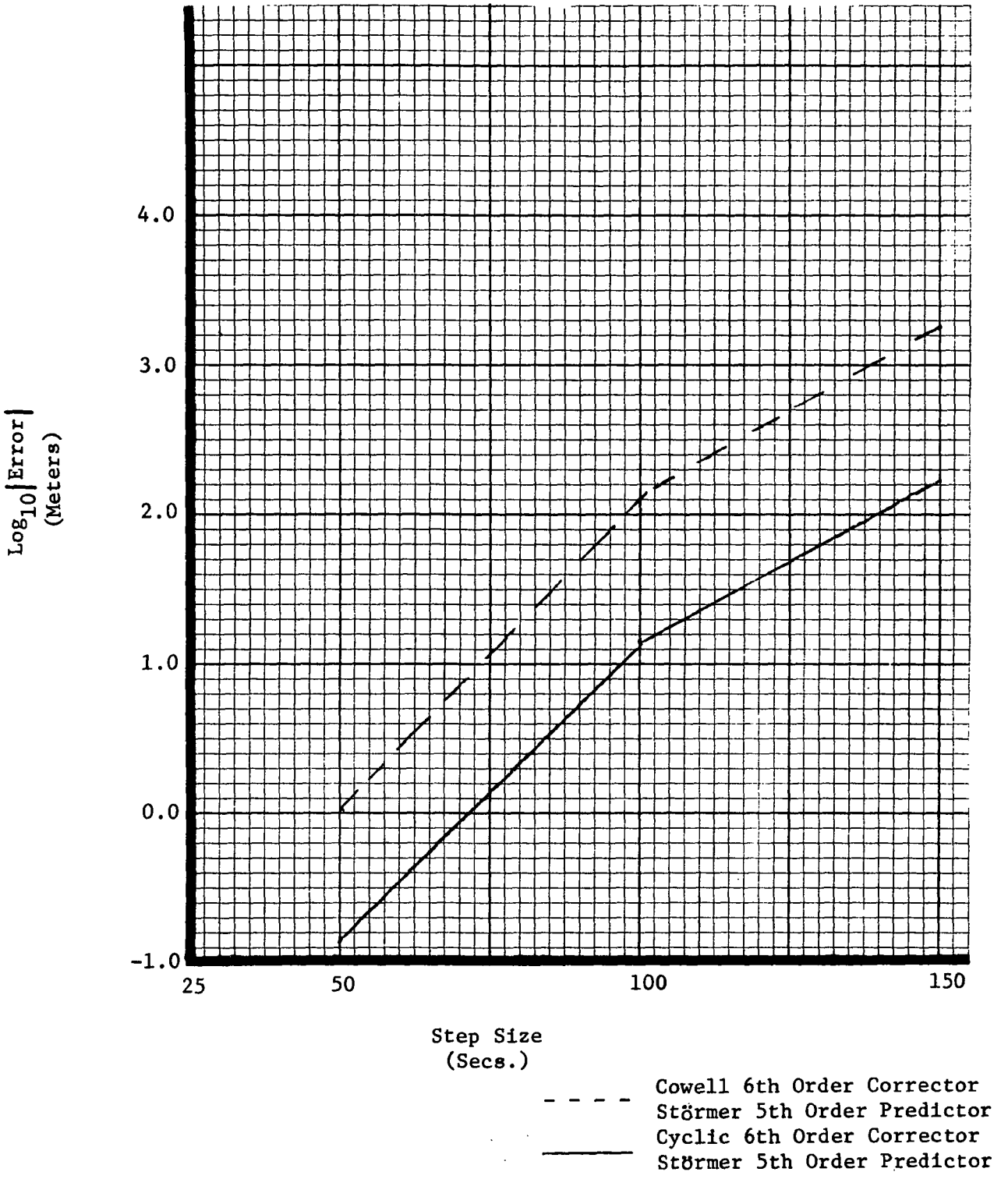

Figure 19. GEOS B Satellite--2-Day Arc
Two-Body Motion
Error vs. Step Size

The results shown in Table 2 are somewhat surprising as the Cowell methods and cyclic method 6 are strongly stable whereas cyclic method 7 is weakly stable. Apparently at this level of accuracy weak stability is not a detriment. Possibly for higher order methods where round-off error is important or for more complex force models weak stability will result in degraded performance. In this case cyclic methods which are strongly stable will be better even though they may have a fraction lower order.

For cyclic method 7 which is weakly stable all of the individual single methods are also weakly stable. In fact these single methods are Henrici's optimal methods for k=4 [8]. This coincidence results because 2k-2 = k+2. This will not occur for k=5.

For k=3, two sets of Class II coefficients were derived. The first is just Cowell's method (method 4). In the second case the two available free parameters were used to increase the order of the method to 4 2/3 (method 5). Two of the individual methods have order 5 and one is the 4th-order Cowell method. The order 5 methods used individually are unstable.

A preliminary conclusion for k=5 is that Class II methods of full order (i.e., individual methods all have order 8) cannot have extraneous roots of zero due to symmetries in the stability equations when the order equations are used. It still may be possible to derive methods in a similar manner to that for k=4.

Future Work

This report leaves many topics on cyclic methods and their relation to other methods open. For the k=4 Class II methods, the remaining degrees of freedom could possibly be used to improve the method. There is still an existence question for k=5 Class II methods and coefficients for Class I and II need to be derived. Much of the work for k=5 is already done since the stability polynomial has been derived. More extensive numerical comparisons of the cyclic methods with other methods are needed to fully evaluate the cyclic methods' potential. In order to do this, however, higher order methods with larger k will have to be derived.

It is recommended that methods with k=6 and k=7 be developed. These methods potentially could have orders of 9-10 and 11-12. Since the algebra involved in obtaining the stability determinant expansion for k=6 and 7 will be formidable, we suggest using either formula manipulation computer programs or numerical methods.

The possibility should also be explored of using the extra degrees of freedom to make $b_k = 0$ yielding some explicit correctors, which would reduce the required number of function evaluations and greatly increase the speed of the method. In fact, just making b_k small may improve the method. Since in practice the step size cannot be zero, the important use of the extra degrees of freedom which should be investigated is to increase the step size interval in which the method remains stable; i.e., choose the parameters such that h may

be chosen nonzero but the roots remain within the unit circle. The necessary theoretical background was begun in paragraph 3.5. Also the effects of putting the cyclic methods in summed form should be investigated. There remain questions on the theoretical side which are important in application of the methods, such as: Is there a cyclic analogue of the corrector iteration convergence theorem (correcting more than once)? What is the meaning of fractional order? Is there a convergence proof for Class II, which will justify the stability criteria we have assumed and give an estimate to the error? Why do Class I methods always exist but Class II for some k, only in special cases? What is the maximum order for a (strongly and weakly) stable method? Is the cyclic method better than others for systems of equations? and, finally, Is there a deep, abstract relation or general theory uniting all the modern methods as there is for the classical ones?

Cyclic methods do seem to be easy to work with since they are composed of classical methods. They appear to be more flexible to use since extra degrees of freedom exist and preliminary results indicate that they will be competitive with the presently used Cowell methods. It is therefore recommended that work on developing cyclic methods be continued.

4. MODIFIED MULTISTEP METHODS

4.1 INTRODUCTION

In conjunction with the current work on problems of orbit computation, modified multistep predictor corrector methods are being explored. A general description of such methods is presented by Gear in [5]. We are developing these techniques for specific application to orbit computation problems. In our approach, position as discussed in section 1 is predicted and corrected by means of Class II methods, which are methods for solving the second-order equation $\ddot{y} = f(t,y)$. Velocity is predicted and corrected by Class I methods, i.e., methods derived from the first order equation $\dot{y} = f(t,y)$.

The potential advantage of the modified methods under development is that they allow increased accuracy over classical methods for a specified number of back points without sacrificing the stability properties of the method. Thus they share, with off-grid and cyclic methods, the property of bypassing the familiar Dahlquist [1] stability results. From a computation point of view both modified and cyclic methods have a potential advantage over off-grid methods of not requiring an extra off-grid calculation.

The off-grid methods discussed in section 2 bypass the Dahlquist criterion by using an extra off-grid calculation. The cyclic methods discussed in section 3 do this by using different correctors in a cyclic fashion. In the modified methods stabilization is accomplished by correcting back points in addition to those currently being calculated. The modified methods have another interesting

property of being derivable by linear transformations of the standard classical methods.

If k^{th} differences are calculated, the methods are exact for polynomials of degree $2k$ for Class I and $2k-1$ for Class II and are strongly stable. In addition they are derived directly by linear transformations of the standard Cowell and Adams-Moulton methods.

To date algorithms have been developed for Class I and Class II methods, and computer programs have been developed for generating the methods' coefficients. These programs are discussed in Appendix C. A subroutine using the modified methods has also been implemented on the GEOSTAR program for integrating the equations of motion with fixed step sizes. This program is described in Appendix D.

Computation results to date are disappointing and further analysis is required to determine if the methods can be improved. The Class I methods agree with those presented by Gear [5], while for Class II there is no standard for comparison and the methods will have to be analysed using the approach suggested in paragraph 4.7.

Presented in the following paragraphs are a discussion of predictor equations, corrector equations, linear transformations, and formulae for generating

modified method coefficients, the coefficients derived to date for Class I and II methods, discussion of the present methods and a discussion of required additional work.

4.2 MODIFIED METHODS

Traditional Class I predictor-corrector methods are defined by the difference equations

Predictor

$$y^{(0)}_{(t_{n+1})} = \sum_{j=1}^{k_1} \alpha_j^* \, y(t_{n-j+1}) + h \sum_{j=1}^{k_2} \beta_j^* \, \dot{y}(t_{n-j+1}) \qquad (4\text{-}1)$$

Corrector

$$y^{(1)}_{(t_{n+1})} = \sum_{j=1}^{k_1} \alpha_j \, y(t_{n-j+1}) + h \sum_{j=0}^{k_2} \beta_j \, \dot{y}^{(1)}_{n-j+1} \qquad (4\text{-}2)$$

where:

t_{n+1}, t_n, t_{n-1} ... are equally spaced points in time,

h = the spacing between these points,

and

$$\dot{y}^{(1)}_{n+1} = f(y^{(0)}_{n+1}, t_{n+1}) \; . \qquad (4\text{-}3)$$

For Class II equations, h is replaced by h^2 and $\dot{y}(t)$ by $\ddot{y}(t)$.

A more general structure for Equations (4-1) and (4-2) can be deduced if we observe that these Equations really consist of operations performed on elements of a space of interpolating polynomials defined by the basis:

$$(y_n,\ y_{n-1},\ \cdots,\ y_{n-k_1+1},\ h\dot{y}_n,\ h\dot{y}_{n-1},\ \cdots,\ h\dot{y}_{n-k_2+1}).$$

Using this concept, equations (4-1) and (4-2) can be reformulated in matrix notation as follows:

Predictor:

$$\bar{y}^{(0)}_{n+1} = B\bar{y}_n \tag{4-4}$$

Corrector:

$$\bar{y}^{(1)}_{n+1} = \bar{y}^{(0)}_{n+1} + \bar{\ell}F. \tag{4-5}$$

$$\bar{y}_n = \begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-k_1+1} \\ h\dot{y}_n \\ \vdots \\ h\dot{y}_{n-k_2+1} \end{bmatrix} \qquad B = \left[\begin{array}{ccc|ccc} \alpha_1^* & \cdots & \alpha_{k_1}^* & \beta_1^* & \cdots & \beta_{k_2}^* \\ 1 & & & & & \\ & \ddots & & & 0 & \\ 0 & & 1\ 0 & & & \\ \hline \gamma_1 & \cdots & \gamma_{k_1} & \delta_1 & \cdots & \delta_{k_2} \\ & & & 1 & & \\ & & & & \ddots & \\ 0 & & & 0 & & 1\ 0 \end{array}\right] \qquad \bar{\ell} = \begin{bmatrix} c_1 \\ \vdots \\ c_{k_1+k_2} \end{bmatrix} \tag{4-6}$$

with

$$\gamma_i = \frac{\alpha_i^* - \alpha_i}{\beta_0} \qquad i = 1, \ldots, k_1$$

$$\delta_j = \frac{\beta_j^* - \beta_j}{\beta_0} \qquad j = 1, \ldots, k_2$$

c_1, c_2 ... are constraints determined by the choice of integration method.

and

$$F = h(\dot{y}_{n+1}^{(0)} - \dot{y}_{n+1}^{(1)}) \tag{4-7}$$

$$h\dot{y}_{n+1}^{(0)} = \sum_{j=1}^{k_1} \frac{\alpha_1^* - \alpha_j}{\beta_0} y(t_{n-j+1}) + h \sum_{j=1}^{k_2} \frac{\beta_i^* - \beta_j}{\beta_0} \dot{y}(t_{n-j+1}). \tag{4-8}$$

In the matrix notation of equations (4-4) and (4-5) choice of $\bar{\ell}$ determines the method. Thus, the traditional Adams methods use $\bar{\ell}^T = [-\beta_0, 0 \ldots 0, -1, 0 \ldots 0]$ along with an appropriate polynomial basis. Gear in [1] has demonstrated the existence of convergent methods which result from other forms of $\bar{\ell}$. In particular, by taking a basis such that $k_1 = k_2 = k$ and choosing $\bar{\ell}$ so that corrections are made to this first k_1 elements of $\bar{y}_{n+1}$ but not to the saved values of the derivatives, he has demonstrated methods which have degree 2k for Class I methods and are strongly stable. These are the modified methods used in our work. The exact coefficients of $\bar{\ell}$ and B are determined in

practice by performing an appropriate linear transformation on the traditional methods (Adams-Moulton for Class I, and Stormer-Cowell for Class II).

To implement the modified methods, it is necessary to determine the coefficients of $\bar{\ell}$ and B. These are obtained in our application by performing the appropriate linear transformations.

$$\bar{y}_M = Q_{AM}\bar{y}_A \,, \tag{4-9}$$

where $\bar{y}_M$ is the basis of the modified method, $\bar{y}_A$ is the basis of the Adams-type methods and Q_{AM} is a change of basis transformation obtained from the appropriate Taylor series relation.

$$\bar{y}_M = \begin{bmatrix} y_n \\ y_{n-1} \\ \\ y_{n-k+1} \\ h\dot{y}_n \\ \\ h\dot{y}_{n-k+1} \\ \\ \end{bmatrix} \qquad \bar{y}_A = \begin{bmatrix} y_n \\ h\dot{y}_n \\ h\dot{y}_{n-1} \\ \\ \\ \\ h\dot{y}_{n-2k+2} \\ \\ \end{bmatrix}$$

Then

$$\bar{\ell}_M = Q_{AM}\bar{\ell}_A \quad \text{and} \quad B_M = Q_{AM}B_AQ_{AM}^{-1} \,, \tag{4-10}$$

where subscripts M and A denote modified and Adams methods respectively. The discussion is also true for Class II type equations except h^2 replaces h and $\ddot{y}$ replaces $\dot{y}$. Also there is one less derivative value for the Class II Adams-type methods.*

In summary, modified methods of orbit computation arise by virtue of making corrections to save orbit position and velocity values in recent passed time as opposed to correcting merely current values, as is done by traditional methods. Modified methods develop a much higher level of local accuracy with the same number of time points entering into the computation.

4.3 COMPUTATION OF MODIFIED METHOD COEFFICIENTS

To implement the modified methods, it is necessary to determine the coefficients of $\overline{\ell}$ and B. These are obtained in our application by performing linear transformations on the traditional type methods. In particular our approach is based on use of Adams-type and Nordsieck-type methods. This involves obtaining values for Q_{AM} and then making the transformations described by equation (4-10).

$$\overline{\ell}_M = Q_{AM}\overline{\ell}_A \text{ and } B_M = Q_{AM}B_AQ_{AM}^{-1}. \qquad (4\text{-}10)$$

*However, for Class II the modified methods are based on quasi-Hermitian interpolation polynomials as discussed in [6] and therefore modified methods will only exist for even k, i.e., k=4, 6, 8, etc. In fact, it turns out that Q_{AM} is singular for k=5, 7, ..., indicating the nonexistence of these methods.

We have developed computer programs which derive Q_{AM},$\bar{\ell}_M$ and B for a specified k. To date we have computed coefficients for modified methods with k=4 and k=6. These methods are exact for polynomials of degree 8 and 12 for Class I and degree 7 and 11 for Class II. The coefficients for these methods are presented in paragraph 4.5. The programs have many self-check features which allow for protecting against errors in input and computation and also provide some estimate of the significance of the computations performed. Since the programs involve performing transformations on Adams, Nordsieck, and the modified methods, it is possible to start with any one method and generate the coefficients of the other. Thus it is possible to generate high-order Nordsieck coefficients from their Adams counterparts. Also the Adams predictor coefficients and all but one of the corrector coefficients can be derived from very simple formulas.

There are several ways to generate the modified method coefficients. These are:

1) Transform Adams methods directly into modified methods.
2) Transform Nordsieck methods directly into modified methods.
3) Transform Adams methods into Nordsieck methods and then transform Nordsieck methods into modified methods.

In the present program we used the third a-proach because it presented less difficulty on the computer than the first approach. The second approach was not feasible because of the unavailability of high-order Nordsieck coefficients.

Now, of course, we can generate them with the programs described in Appendix C. Also the third approach greatly facilitates the checking of various types of errors that could occur. The change-of-basis transformation matrix Q_{AM} is given by

$$Q_{AM} = Q_{NM}Q_{NA}^{-1} , \tag{4-11}$$

where Q_{NM} is the change-of-basis transformation from the Nordsieck to the modified method, and Q_{NA} is the change-of-basis transformation from the Nordsieck to the Adams methods. Once Q_{AM} is calculated, the modified method coefficients are obtained by application of equation (4-10).

The transformations Q_{NM} and Q_{NA} are obtained by using the appropriate Taylor series representations.

The ij^{th} element of Q_{NA} is given by:

Class I

$$Q_{NA}^{ij} \begin{cases} 1 \text{ for } j=0,\ i=0 \\ 0 \text{ for } j=1,\ldots,2k-1,\ i=0 \\ 0 \text{ for } j=0,\ i=1,\ldots,\ 2k-1 \\ 1 \text{ for } j=1,\ i=1 \\ 0 \text{ for } j=2\ldots,\ 2k-1,\ i=1 \\ j(-i+1)^{j-1} \text{ for } j=1,\ldots,\ 2k-1,\ i=2\ldots,\ 2k-1 \end{cases} \tag{4-12}$$

Class II

$$Q_{NA}^{ij} \begin{cases} 1 \text{ for } j=0,\ i=0 \\ 0 \text{ for } j=0,\ i=2,\ldots,\ 2k-1 \\ 0 \text{ for } j=1,\ldots,\ 2k-1,\ i=0 \\ (-1)^j \text{ for } j=0,\ldots,\ 2k-1,\ i=1 \\ 0 \text{ for } j=1,\ 3\ldots,\ 2k-1,\ i=2 \\ 2 \text{ for } j=2,\ i=2 \\ 0 \text{ for } j=1,\ i=3,\ldots,\ 2k-1 \\ j(j-1)(-i+2)^{j-2} \text{ for } j=2,..,\ 2k-1,\ 1=3,\ldots,\ 2k-1 \end{cases} \qquad (4\text{-}13)$$

The ij^{th} element of Q_{NM} is given by:

Class I

$$Q_{NM}^{ij} \begin{cases} 1 \text{ for } j=0,\ i=0 \\ 0 \text{ for } j=1,\ldots,\ 2k-1,\ i=0 \\ (-i)^j \text{ for } j=0,\ldots,\ 2k-1,\ i=1,\ldots,\ k-1 \\ 0 \text{ for } j=0,\ i=k,\ldots,\ 2k-1 \\ 1 \text{ for } j=1,\ i=k \\ 0 \text{ for } j=2,\ldots,\ 2k-1,\ i=k \\ j(-i+k)^{j-1} \text{ for } j=1,\ldots,\ 2k-1,\ i=k+1,\ldots,\ 2k-1 \end{cases} \qquad (4\text{-}14)$$

Class II

$$Q_{NM}^{ij} \begin{cases} 1 \text{ for } j=0,\ i=0 \\ 0 \text{ for } j=1,\ldots,\ 2k-1,\ i=0 \\ (-i)^j \text{ for } j=0,\ldots,\ 2k-1,\ i=1,\ldots,\ k-1 \\ 0 \text{ for } j=0,\ 1,\ i=k,\ldots,\ 2k-1 \\ 2 \text{ for } j=2,\ i=k \\ 0 \text{ for } j=3,\ldots,\ 2k-1,\ i=k \\ j(j-1)(-i+k)^{j-2} \text{ for } j=2,\ldots,\ 2k-1,\ i=k+1,\ldots,\ 2k-1 \end{cases} \tag{4-15}$$

4.4 COMPUTATIONAL PROCEDURES

Using equations (4-12), (4-13), (4-14) and (4-15), it is possible to generate Q_{NA}, Q_{NA}^{-1}, Q_{NM}, and Q_{AM}^{-1} for a specified k. The coefficients for the Adams-Moulton and the Stormer-Cowell methods could be input, and B_A and $\bar{\ell}_A$ could be formed according to equation (4-6). Then B_M and $\bar{\ell}_M$ could be computed using equation (4-10).

The matrix B_M can also be computed directly from Nordsieck's method by using Q_{NM} and the Nordsieck B_N. This matrix is simply a triangular matrix with binomial coefficients for the nonzero elements. The ij^{th} element of B_N for both Class I and Class II is given by

$$B_N^{ij} \begin{cases} 0 & j < i \\ & i, j=0,\ldots, 2k-1. \\ \binom{j}{i} & j \geq i \end{cases} \qquad (4\text{-}16)$$

Computing B_M both ways provides a useful check on the computations and on the method used.

The actual procedures implemented use the formulas for B_N (4-16) and Q_{NM} (4-14), (4-15) allowing the computation of B_M from these simple formulae, without any prior knowledge of either the Adams-Moulton or the Störmer-Cowell coefficients. In this case it is only necessary to input one coefficient each from the Cowell and Moulton correctors to generate B_M and $\overline{\ell}_M$. This procedure is described in Appendix C and is the approach used to generate the coefficients in paragraph 4.5

Though not used in our work, it is possible to go even further and use relations similar to those presented in [11,p 30] to derive the remaining coefficients in the corrector formula through the application of B_N^{-1}. With this latter feature it is possible to have a single fairly simple program which will be able to derive the coefficients of the Adams-Moulton, the Störmer-Cowell, the Nordsieck Class I and II, and the modified multistep methods simply by inputting k and the desired method. The programs described in Appendix C could be easily modified to have this unique feature (i.e., they practically have it now).

4.5 COEFFICIENTS FOR CLASS I AND II MODIFIED METHODS

To date 4-step and 6-step modified methods for Class I and II have been developed and are implemented on the GEOSTAR program. Using the notation of equation (4-6) we have

$$B_M = \begin{bmatrix} \alpha_1^* & \cdots & \alpha_k & \beta_1^* & \cdots & \beta_k^* \\ 1 & & & & & \\ & \ddots & 0 & & & \\ & & 1 & & 0 & \\ \hline \gamma_1 & & \gamma_k & \delta_1 & & \delta_k \\ & 0 & & 1 & & \\ & & & & \ddots & 0 \\ & & & & 1 & 0 \end{bmatrix} \qquad \bar{\ell}_M = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4-17}$$

k=4 Modified Method

Class I

$\alpha_1^* = -42.66$ ---- $\beta_1^* = 16.0$ $c_1 = -.3042245370370370$

$\alpha_2^* = -36.0$ $\beta_2^* = 72.0$ $c_2 = .1136739417989418 \times 10^{-1}$

$\alpha_3^* = 64.0$ $\beta_3^* = 48.0$ $c_3 = -.2901785714285714 \times 10^{-2}$

$\alpha_4^* = 15.66$ ---- $\beta_4 = 4.0$ $c_4 = .1579034391534392 \times 10^{-2}$

$c_5 = -1.0$

$\gamma_1 = -151.11$ ---- $\delta_1 = 50.66$ ---- $c_6 = 0.0$

$\gamma_2 = -150.0$ $\delta_2 = 264.0$ $c_7 = 0.0$

$\gamma_3 = 240.0$ $\delta_3 = 184.0$ $c_8 = 0.0$

$\gamma_4 = 61.11$ ---- $\delta_4 = 15.66$ ----

Class II

$\alpha_1^* = -16.0$ $\beta_1^* = 2.66$ ---- $c_1 = -.7134589947089947 \times 10^{-1}$

$\alpha_2^* = 34.0$ $\beta_2^* = 14.66$ ---- $c_2 = 0.0$

$c_3 = -.3654100529100529 \times 10^{-2}$

$\alpha_3^* = -16.0$ $\beta_3^* = 2.66$ ---- $c_4 = -.3141534391534392 \times 10^{-2}$

$\alpha_4^* = -1.0$ $\beta_4^* = 0.0$ $c_5 = -1.0$

$c_6 = 0.0$

$c_7 = 0.0$

$\gamma_1 = -240.0$ $\delta_1 = 24.0$ $c_8 = 0.0$

$\gamma_2 = 480.0$ $\delta_2 = 194.0$

$\gamma_3 = -240.0$ $\delta_3 = 24.0$

$\gamma_4 = 0.0$ $\delta_4 = -1.0$

k=6 Modified Method

Class I

$\alpha_1^* = -128.4$ $\beta_1^* = 36.0$ $c_1 = -.2742655400315991$

$\alpha_2^* = -750.0$ $\beta_2^* = 450.0$ $c_2 = .5924056412337662 \times 10^{-2}$

$\alpha_3^* = -400.0$ $\beta_3^* = 1200.0$ $c_3 = -.8617935722970445 \times 10^{-3}$

$\alpha_4^* = 825.0$ $\beta_4^* = 900.0$ $c_4 = .2449104554139276 \times 10^{-3}$

$\alpha_5^* = 426.0$ $\beta_5^* = 180.0$ $c_5 = -.1123681006493506 \times 10^{-3}$

$\alpha_6^* = 28.4$ $\beta_6^* = 6.0$ $c_6 = .7722834328737107 \times 10^{-4}$

$c_7 = -1.0$

$c_8 = 0.0$

$c_9 = 0.0$

$c_{10} = 0.0$

$c_{11} = 0.0$

$c_{12} = 0.0$

$\gamma_1 = -536.76$ $\delta_1 = 140.4$

$\gamma_2 = -3412.5$ $\delta_2 = 1980.0$

$\gamma_3 = -1960.0$ $\delta_3 = 5480.0$

$\gamma_4 = 3780.0$ $\delta_4 = 4185.0$

$\gamma_5 = 1995.0$ $\delta_5 = 846.0$

$\gamma_6 = 134.26$ $\delta_6 = 28.4$

Class II

$\alpha_1^* = -\ 59.12658227848101$ $\beta_1^* = 4.898734177215190$ $c_1 = -\ .6107264986171236 \times 10^{-1}$

$\alpha_2^* = -\ 121.3291139240506$ $\beta_2^* = 76.10126582278481$ $c_2 = 0.0$

$\alpha_3^* = 362.9113924050633$ $\beta_3^* = 204.8354430379747$ $c_3 = -\ .2067782237053010 \times 10^{-2}$

$\alpha_4^* = -121.3291139240506$ $\beta_4^* = 76.10126582278481$ $c_4 = -\ .1780240400032067 \times 10^{-2}$

$\alpha_5^* = -\ 59.12658227848101$ $\beta_5^* = 4.898734177215190$ $c_5 = -\ .1845983393462562 \times 10^{-2}$

$\alpha_6^* = -\ 1.0$ $\beta_6^* = 0.0$ $c_6 = -\ .1832085738335736 \times 10^{-2}$

$c_7 = -\ 1.0$

$c_8 = 0.0$

$c_9 = 0.0$

$c_{10} = 0.0$

$c_{11} = 0.0$

$c_{12} = 0.0$

$\gamma_1 = -\ 1152.341772151899$ $\delta_1 = 71.12658227848101$

$\gamma_2 = -\ 2807.088607594937$ $\delta_2 = 1494.873417721519$

$\gamma_3 = 7918.860759493671$ $\delta_3 = 4286.455696202532$

$\gamma_4 = -\ 2807.088607594937$ $\delta_4 = 1494.873417721519$

$\gamma_5 = -\ 1152.341772151899$ $\delta_5 = 71.12658227848101$

$\gamma_6 = 0.0$ $\delta_6 = -\ 1.0$

4.6 ORDER OF THE METHODS DEVELOPED

As discussed in paragraph 2.6.1, traditional multistep methods can be expressed in operator form. The Class II operator is given by

$$L(x,y(x),h) = a_k(x+kh) + a_{k-1}y(x+(k-1)h) \ldots a_o y(x) + h^2\{b_k y''(x+kh) + b_{k-1}y''(x+(k-1)h) \ldots b_o y''(x)\} \quad (4\text{-}18)$$

This in turn can be expanded by the Taylor series in terms of a polynomial in h.

$$L(x,y(x),h) = C_0 y(x) + C_1 y'(x)h + C_2 y''(x)h^2 \ldots C_p y^{(p)}_{(x)} h^p \quad (4\text{-}19)$$

The largest value of p such that $C_0, C_1, \ldots, C_p = 0$, and $C_{p+1} \neq 0$ represents the degree of the polynomial for which the method is exact. The number p-1 for Class II methods and p for Class I methods is usually referred to as the order of the method [8]. The importance of the value of p is that it gives an estimate of the local truncation error for the method. This fact was used in paragraph 2.6 for the discussion of variable step procedures.

In general the Class II equations for C_p, p=0,1, .. are as follows

$$C_0 = a_0+a_1+a_2 \ \ldots \ a_k$$

$$C_1 = \ \ a_1+2a_2 \ \ldots \ ka_k$$

$$C_2 = \frac{1}{2!}(a_1+2^2a_2 \ldots \ k^2a_k) + (b_0+b_1 \ \ldots \ b_k)$$

$$\vdots$$

$$C_p = \frac{1}{p!}(a_1+2^pa_2 \ldots \ k^pa_k) + \frac{1}{(p-2)!}(b_1 \ \ldots \ k^{p-2}b_k) \qquad (4\text{-}20)$$

Similar relations exist for Class I methods.

Modified methods also satisfy relations similar to (4-20). However, modified methods use k different correctors* instead of the single corrector used by the classical methods. Since they use k correctors (one for each back point correction), there are k different sets of p equations of the form (4-20) that must be satisfied. In this sense the modified methods are similar to the cyclic methods discussed in section 3. Analysis of this connection merits further investigation.

To illustrate how the k set of order equations for the modified methods are obtained, it is useful to look at a simple case. Equations (4-5), (4-6), and (4-7) give the form of the modified correctors in matrix notation. This is

*Predictors of the method are analyzed in the standard manner.

$$\bar{y}^{(1)}_{n+1} = \bar{y}^{(0)}_{n+1} + \bar{\ell}(h^2\ddot{y}^{(0)}_{n+1} - h^2\ddot{y}^{(1)}_{n+1}) \tag{4-21}$$

Expanding this expression for the correction made to the first term (i.e. y_{n+1}) gives

$$y^{(1)}_{n+1} = \sum_{i=1}^{k} y_{n-i+1}(\alpha^*_i + c_1\gamma_i) + h^2\left(-c_1\ddot{y}^{(0)}_{n+1} + \sum_{i=1}^{k}\ddot{y}_{n-i+1}(\beta^*_i + c_1\delta_i)\right) \tag{4-22}$$

Then by inspection the "1st corrector" must satisfy the order equations (4-20) with the following coefficients

$$a_k = -1 \qquad b_k = -c_1$$

$$a_{k-1} = \alpha^*_1 + c_1\gamma_1 \qquad b_{k-1} = \beta^*_1 + c_1\beta_1$$

$$\vdots \qquad\qquad \vdots$$

$$a_0 = \alpha^*_k + c_1\gamma_k \qquad b_0 = \beta^*_k + c_1\delta_k$$

The same procedure can be repeated to determine the proper set of coefficients for the "other" correctors of the method. Thus the "2nd corrector" is given by

$$y^{(2)}_{n+1} = y^{(1)}_{n+1} + c_2\left(\sum_{i=1}^{k}\gamma_i y_{n-i+2} + h^2\sum_{i=1}^{k}\delta_i\ddot{y}_{n-i+2} - c_2h^2\ddot{y}^{(0)}_{n+2}\right) \tag{4-23}$$

which when expanded becomes

$$y^{(2)}_{n+1} = c_2\gamma_1 y_{n+1} + \sum_{i=1}^{k} y_{n-i+1}(\alpha^*_i + c_1\gamma_i + c_2\gamma_{i+1}) + y_{n-k+1}(\alpha^*_k + c_1\gamma_k)$$

$$+ h^2\left(-c_2\ddot{y}^{(0)}_{n+2} - c_1\ddot{y}^{(0)}_{n+1} + c_2\delta_1\ddot{y}_{n+1} + \sum_{i=1}^{k-1} \ddot{y}_{n-i+1}(\beta^*_i + c_1\delta_i + c_2\delta_{i+1})\right.$$

$$\left. + \ddot{y}_{n-k+1}(\beta^*_k + c_1\delta_k)\right) \qquad (4\text{-}24)$$

The "2nd" corrector must satisfy the order equations with the following coefficients

$a_k = 0$ $\qquad$ $b_{k+1} = -c_2$

$a_k = c_2\gamma_1 - 1$ $\qquad$ $b_k = -c_1 + c_2\delta_1$

$a_{k-1} = \alpha^*_1 + c_1\gamma_1 + c_2\gamma_2$ $\qquad$ $b_{k-2} = \beta^*_1 + c_1\delta_1 + c_2\delta_2$

. $\qquad$.

. $\qquad$.

. $\qquad$.

$a_1 = \alpha^*_{k-1} + c_1\gamma_{k-1} + c_2\gamma_k$ $\qquad$ $b_1 = \beta^*_{k-1} + c_1\delta_{k-1} + c_2\delta_k$

$a_0 = \alpha^*_k + c_1\gamma_k$ $\qquad$ $b_0 = \beta_k + c_1\delta_k$

Note that the corrector with respect to the order equations behaves as though it had k+1 back points. In general, the m^{th} corrector will behave as though it had k+m-1 back points and satisfy the order equations (4-20) with the following coefficients

$$s=m-1, \text{ for } m=1,\ldots,k$$

$$a_{k+s} = 0$$

$$a_{k+s-1} = c_s\gamma_1+c_{s+1}\gamma_2$$

$$\vdots$$

$$a_k = c_2\gamma_1+c_3\gamma_2 \cdots c_{s+1}\gamma_s-1$$

$$a_{k-1} = \alpha_1^*+c_1\gamma_1+c_2\gamma_2 \cdots c_{s+1}\gamma_{s+1}$$

$$a_{k-2} = \alpha_2^*+c_1\gamma_2+c_2\gamma_3 \cdots c_{s+1}\gamma_{s+2}$$

$$\vdots$$

$$a_s = \alpha_{k-s}^*+c_1\gamma_{k-s} \cdots c_{s+1}\gamma_k$$

$$a_{s-1} = \alpha_{k-s+1}^*+c_1\gamma_{k-s+1} \cdots c_s\gamma_k$$

$$\vdots$$

$$a_1 = \alpha_{k-1}^*+c_1\gamma_{k-1}+c_2\gamma_k$$

$$a_0 = \alpha_k^*+c_1\gamma_k \qquad (4\text{-}25)$$

$$b_{k+s} = -c_{s+1}$$

$$b_{k+s-1} = -c_s + c_{s+1}\delta_1$$

$$\vdots$$

$$b_k = -c_1 + c_2\delta_1 \cdots c_{s+1}\delta_s$$

$$b_{k-1} = \beta^*_1 + c_1\delta_1 + c_2\delta_2 \cdots c_{s+1}\delta_{s+1}$$

$$\vdots$$

$$b_s = \beta^*_{k-s} + c_1\delta_{k-s} + c_2\delta_{k-s+1} \cdots c_{s+1}\delta_k$$

$$b_{s-1} = \beta^*_{k-s+1} + c_1\delta_{k-s+1} \cdots c_s\delta_k$$

$$\vdots$$

$$b_1 = \beta^*_{k-1} + c_1\delta_{k-1} + c_2\delta_k$$

$$b_0 = \beta_k + c_1\delta_k \qquad (4\text{-}26)$$

The orders of the developed Class II modified methods for k=4 and k=6 have been evaluated using equations (4-20), (4-25), and (4-26). For k=4, for all four correctors and the predictor, $C_0,\ldots,C_7=0$, $C_8 \neq 0$. For k=6, for all six correctors and the predictor $C_0,\ldots,C_{11}=0$, $C_{12} \neq 0$. The methods are exact for 7th and 11th degree polynomials for k=4 and k=6, respectively.

4.7 RESULTS TO DATE

Results to date for the modified methods are disappointing. Table 4 presents computational results for the k=4 and k=6 Class II modified methods. The GEOSB satellite with a two-body force model was used. As can be seen from the table 4, the modified k=4 (order 6) method is inferior to the cyclic method of order 6 but superior to the Cowell method of order 6. The k=6 method is inferior at small step sizes (h=50 sec.) but is superior to both the modified k=4 and cyclic 6th order methods at larger step sizes (h=100 sec.).

The reason why the k=6 method performs worse at small step sizes than at larger step sizes is unclear. This is a similar phenomenon to that observed for off-grid methods where in many situations smaller step sizes do not necessarily produce better results. Part of this effect is no doubt due to roundoff error, however a large part is probably due to the structure of the method itself.

At h=100 secs. the k=6 method is inferior to either the off-grid or the presently used Cowell method (e.g. error for 11th order Cowell is .03 meters at 100 secs.). There are three possibilities to be considered. These are either, the modified methods are presently inferior but can be improved within the framework of the present theory, the methods as presently derived are not optimal and superior methods can be developed, or the methods are inherently less optimal than Cowell or off-grid methods for orbit computation problems.

Table 4. Results for Modified Class II Methods

GEOSB Satellite--Two-Body Forces

		PROPAGATED ERROR* (METERS)			
Step Size (Secs.)	Length of Arc (Hours)	Modified k=4	Modified k=6	Cowell 6th Order	Cyclic 6th Order
50	2	.0019	.013	.018	.003
	12	.032	.40	.12	.02
	26	.12	1.6	.33	.08
100	2	.23	.0044	1.37	.21
	12	3.77	.15	11.7	1.98
	26	13.9	.64	33.5	6.05

*Error measured as difference between approximated and analytic solutions for the x coordinate of position.

The present method could be made more accurate by improving roundoff properties. This could be done by altering the computational scheme and if possible developing the method in summed form.

Another significant improvement would result from allowing, where necessary, more than one corrector iteration. The present algorithm is programmed for only one corrector iteration. It was programmed in this manner to keep the number of function evaluations to a minimum. However the Cowell 11th order method averages 1.73 corrector iterations per step for a step size of 100 seconds on this same orbit. The error in this case was .03 meters vs .64 meters for the k=6 modified method. It is possible that allowing for more than one iteration would considerably improve the accuracy of the modified k=6 method. Unfortunately it would also increase the number of function evaluations and subsequently computation time. The effect of additional corrections on accuracy can be easily tested since incorporating this feature into the present program would only require minor changes.

The next question is whether better modified methods exist and can be developed with either improved accuracy or stability properties. The present methods apparently satisfy the order equations developed in the previous section. This in itself does not ensure uniqueness. For example the order equations are nonlinear in terms of $\overline{\ell}$ and, without further study, it cannot be stated with certainty whether they can be satisfied by one or several values of $\overline{\ell}$. In addition to the order equations the stability criterion in [5]

must also be satisfied. It is recommended that additional analysis be made which involves solving the difference equations for the error term and using the stability criterion in [5] to ensure that the eigenvalues of the appropriate coefficient matrix are within the unit circle. Until this is done it cannot be known for sure whether the methods developed are stable and valid. The Class I methods developed are equivalent to those presented by Gear in [5].* Final conclusions as to the usefulness of the modified methods for orbit computations cannot be made until this analysis is performed.

*Gear presents results only through k=4. To this step size the methods are equivalent.

5. DEVELOPMENT OF IMPROVED MULTISTEP INTEGRATION METHODS

5.1 INTRODUCTION

We are currently developing three different types of high-order multistep integration methods for application to problems of orbit computation (i.e. off-grid, cyclic and modified multistep methods). As discussed in the previous sections, these methods all have the property of not being limited by the familiar Dahlquist [1] stability results as are the traditional Gauss-Jackson and Cowell methods. They thus allow for higher order than these classical methods without sacrificing their stability properties.

For each of these three approaches it may be possible to develop what could be called "stabilized methods." These are methods in which either the maximal order constraint has been dropped or the method is otherwise modified to provide for the use of large integration step sizes. All three methods currently being evaluated for NASA can probably be significantly improved by some type of stabilization procedure. This also may be true for the presently used Cowell method.

In the following paragraphs the concepts of accuracy and stability are discussed. A general procedure for developing optimum methods is presented. The application of this technique to cyclic methods and the results of preliminary numerical experiments are discussed. It is strongly recommended that further work be done on developing stabilized methods.

5.2 ACCURACY AND STABILITY

Traditional Class I multistep methods can be expressed in the form

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} \cdots a_k y_{n-k} + h\left\{b_o y'_n + b_1 y'_{n-1} \cdots b_k y'_{n-k}\right\} \tag{5-1}$$

Such methods are used to solve the initial value problem.

$$z' = f(x,z) \quad z'(a) = c \tag{5-2}$$

where $y'_n = f(x_{n+1}, y_n)$ and y_n are derived from expression (5-1) given an appropriate set of starting values. The integration step size is h and k is the number of back points used in the approximation procedure.

The difference equation (5-1) is only an approximation of the differential equation (5-2). Therefore the true solution will only approximately satisfy (5-1).

$$z_n = a_1 z_{n-1} \cdots a_k z_{n-k} + h\left\{b_o z'_n \cdots b'_k z'_{n-k}\right\} + R_n \tag{5-3}$$

The error at each integration step is given by

$$e_n = z_n - y_n \tag{5-4}$$

From the mean value theorem

$$z'(x) - y' = \frac{\partial f(x,\theta)}{\partial z}(z(x)-y) = A(\theta)e$$

subtracting (5-1) from (5-3) results in

$$e_n = a_1 e_{n-1} + a_2 e_{n-2} \cdots + h\left\{b_o A(\theta_o) e_n \cdots b_k A(\theta_k) e_{n-k}\right\} + R_n \tag{5-6}$$

or rewriting (5-6) gives

$$e_n(1 - b_o A(\theta_o)) = e_{n-1}(a_1 + hA(\theta_1)) \ldots e_k(a_k + hA(\theta_k)) + R_n \qquad (5\text{-}7)$$

Equation (5-7) is a linear nonhomogeneous difference equation. The solution will be represented by two parts. One part being the solution to the homogeneous equation given by (5-8)

$$-e_n(1 - b_0 A(\theta_o)) + e_{n-1}(a_1 + hA(\theta_1)) + \ldots e_k(a_k + hA(\theta_k)) = 0 \qquad (5\text{-}8)$$

and the other part is the solution of the nonhomogeneous term represented by the function R_n.

The term R_n is the error resulting from approximating the differential equation (5-2) with the difference method (5-1). It is the local discretization error or the error that occurs with one step of the method. A simple expression for R_n is readily derived by expanding (5-3) in terms of a Taylor series. If the method is of order p then

$$R_n = C_{p+1} h^{P+1} z^{(p+1)}_{(x_{n-k})} + 0(h^{p+2}) \qquad (5\text{-}9)$$

where

$$z^{(p+1)}_{(x_{n-k})} = \left. \frac{d^{p+1} z(r)}{dz(r)^{p+1}} \right|_{r = x_{n-k}} \qquad (5\text{-}10)$$

$$C_{p+1} = \frac{a_1 + 2^{p+1}a_2 \ldots k^{p+1}a_k}{(p+1)!} - \frac{b_1 + 2^{p}b_2 \ldots k^{p}b_k}{p!} \qquad (5\text{-}11)$$

Alternate formulations of R_n are possible which only possess h^{p+1} terms using the influence function discussed by Hamming [12], and Henrici [8]. In actual practice this approach will be utilized when appropriate. Expression (5-9) is sufficient for purposes of illustration and has been used for most of the examples presented in the discussion.

The homogeneous solution to equation (5-8) is of the form

$$e_n = c_1(\rho_1)^n + c_2(\rho_2)^n \ldots c_k(\rho_k)^n \qquad (5\text{-}12)$$

where $\rho_1, \ldots, \rho_k$ are roots of the characteristic polynomial.

$$(1 - b_0 hA(\theta_o))^k - (a_1 + hA(\theta_1))\rho^{k-1} \ldots (a_k + hA(\theta_k)) = 0 \qquad (5\text{-}13)$$

and the coefficients $c_1, \ldots, c_k$ are determined from initial conditions of the difference equation.

From (5-12) it is clear that the error e_n will grow with out bound as n gets large unless all the roots ρ_i, i = 1 ... k, are less than or equal to one (i.e., $|\rho_i| \leq 1$). This later concept is referred to as a stability requirement. That is, the roots of (5-13) must be such that the error e_n decays rather than

grows with each intergration step. In addition to this general stability requirement, most practical methods also satisfy the convergence criteria discussed by Henrici. [8] This result states that as $h \to 0$, the modulus of no root of (5-13) exceeds 1, and that the roots of modulus 1 are simple. The method also must be consistent (i.e. one of the roots of (5-13) as $h \to 0$ is 1).

To summarize the discussion so far, we have identified two types of error terms. These are, in the absence of round-off errors,

a) Accuracy of the approximating difference equation

b) Stability of the approximating method

The accuracy of the approximating method is a function of the h^{p+1} terms, a set of coefficients which are a function of the method, and a high-order derivative $z^{(p+1)}_{(\xi)}$ of the function. In general, we can expect that high-order methods will have smaller truncation errors than low-order ones.

The stability of the method depends on the coefficients of the method, the integration step size, and the properties of the function as represented by $\frac{\partial z'}{\partial z}$ being integrated. Instability in essence results when the integrating method causes a systematic error which grows without bound. A stable method is one in which the coefficients are such that the errors from one step cancel those of a previous step thereby preventing divergence of the process.

5.3 DEVELOPMENT OF OPTIMUM METHODS

The effects of round-off errors and techniques to handle them for the methods under consideration will be treated in another paper. Generally speaking, the cyclic methods should be amenable to many of the commonly used approaches* while modified and off-grid methods will probably require modification. Double precision arithmetic will, of course, be adequate for many applications.

Other than roundoff, the two major sources of error are instability and truncations. Stability is governed by the roots of (5-13) and truncation is governed by (5-9) or an equivalent expression. These two expressions form the basic tools for optimizing integration methods. By optimization it is meant to maximize, for a specified differential equation $Z'=f(X,Z)$, the integration step size h which is compatible with specified accuracy requirements.

Ideally, this procedure is one of minimizing truncation error while maximizing the stability region. However, it is impossible for classical methods** to achieve both of these extremes.[8] Generally, as h is increased, both accuracy and stability decrease. However, in practical applications, achieving both these extremes is unnecessary.

---

* Summed form, etc.

** There has been little theoretical work in the area of what might be achievable with cyclic methods using the apparent multitude of available parameters.

Instead, when a method fails for a specified value of h, it must be determined whether the failure was attributable to truncation error or stability. Then using (5-9), (5-11), and (5-13), the coefficients of the method are changed to either increase the stability region or reduce the truncation error. In the case of low-order methods, the accuracy can be increased. For high-order methods, stability usually needs improvement.

In most conventional methods, where there are only a few manipulatable parameters, stability is increased at the expense of accuracy, and accuracy is increased at the expense of a reduced stability region. However, the net result will be a more efficient method for a specified class of differential equations.

This last point should not be overlooked. In the case of conventional methods, we are in essence optimizing the integration method for a specified class of problems. These optimum methods will perform better for that class of equation than a more general method. This approach should be particularly applicable to orbit computation problems since the same set of basic equations are used for most computations.

A summary of the basic approach can be stated as follows:

A) A given conventional integration method, defined by (5-1) is specified by a set of 2K + 2 parameters (a_o ,...,a_k,b_o, ...,b_k).

B) The order P of the method, which effects the truncation error, is specified by set of P + 1 equations, linear in the coefficients, similar in form to (5-11). Thus P + 1 parameters are specified.

C) In general, methods of order $k + 1$ are used, although it is possible to achieve order $k + 2$ when K is even. An order of $k + 1$ requires specification of $k + 2$ parameters.

D) One parameter is specified as a result of the normalization of (5-1).

E) If maximum-order methods are used, there are K-1 free parameters which can be manipulated using (5-13) or (5-9). to either increase stability or reduce truncation error. In addition to satisfying the stability criteria for finite h, these parameters must also satisfy it as $h \rightarrow o$.

F) Additional parameters can be used if the order of the method is reduced. One parameter is made available for each reduction of order.

So far the above discussion has neglected an analysis of propagated error. The accumulated error after n steps is, of course, our real concern. Therefore it should be verified that the above approach of reducing truncation error and increasing stability regions will result in smaller propagated errors.

An expression for the propagated error after n steps can be derived by recursively solving equation (5-7). This has been done by Nigro who presents an expression for the propagated error. His results verify the validity of our approach [13].

Another analysis of the resultant propagated error for a specified range of integration is presented by Henrici [8]. Henrici has developed expressions for a propagated error bound. We have explored the feasibility of developing optimum methods using this bound. For practical application such analysis is very cumbersome. However an analysis of Henrici's results will be of theoretical value when we optimize cyclic methods.

5.4 ANALYSIS OF THE CHARACTERISTIC POLYNOMIAL

In working with equation (5-13), two types of problems arise. One is the problem of analyzing roots of a high-order polynomial. Many authors have developed results useful for this purpose. Reference [14] is representative of this type of analysis. Nigro has derived an explicit expression for the maximum h for which a given method is stable [13]. The local behavior of roots about a specified point can be studied by power-series expansions. Finally, where necessary, numerical techniques can be used.

The other more basic problem is deriving appropriate values for $A(\theta_o)$, ..., $A(\theta_k)$ for use in the analysis. Hamming states that for a large class of

problems $\frac{\partial f}{\partial y}$ varies very slowly [12]. This, at least, allows us to set $A(\theta_o) = A(\theta_1) = \ldots A(\theta_k) = A$. However, we still have to pick a value for A. One option is to define some average value of A for the specified range of integration.

Nigro uses the critical eigenvalue of the Jacobian matrix associated with a system of equations [13]. For a single nonlinear equation, this eigenvalue will be the same as our $A(\theta_o) \ldots A(\theta_k)$. There is no formal interpretation of a critical eigenvalue for a nonlinear equation. However, we could heuristically interpret this to mean the $\max_X \left| \frac{\partial f(X,Z)}{\partial Z} \right|$. If we use the entire range of function definition and if this function is single-valued, this is equivalent to the Lipschitz constant. An alternate to this might be to linearize the differential equation using a Taylor series expansion and use the first term to define A.

Rather than pick a constant value for A, we could allow it to vary as the integration procedure progresses. Thus, the coefficients of the method are not constant but rather vary as a function of $\frac{\partial f(X,Z)}{\partial Z}$. One such approach is presented by Rahme [15]. Rahme's approach merits further consideration.

5.5 APPLICATION TO CYCLIC METHODS

All three methods (off-grid, modified, and cyclic) and also the conventional methods can probably be improved by applying the techniques described in the previous section. The cyclic methods are especially interesting since modifications of the present method might provide additional free parameters without requiring a reduction in the order of the system.

The cyclic methods employ M different correctors cyclically. Hansen in [4] develops methods for when $M = k$. This approach is described in detail in section 3 where the stability criterion for nonzero step size has also been developed. These methods are of order $2k-1$ for Class I equations and various orders for Class II equations. For the Class I case there is one free parameter which can be used to improve the system. If M is made greater than k, additional free parameters will be available for improving the method. The total number of available parameters is $M-k+1$. In general there should be some limit to the value of adding additional correctors to the cycle. However, at present there is no theory to indicate what such a limit is. Basically the following options are available:

A) Make M>k and use the additional M+1-k parameters to reduce truncation error, and increase stability by adapting the previously discussed analysis to the cyclic formulations.

B) Make M>k and use the additional M+1-k parameters to develop M+1-k explicit methods. This will eliminate M+1-k function evaluations per M steps. Computing time is reduced by $\frac{M+1-k}{M}$ over the all-corrector method (of course, accuracy and stability will change).

C) Keep M=k, but reduce the order of the system thus obtaining additional parameters which can be used to stabilize the system or to develop an all-explicit system.

D) Make M>k, reduce the order of the system so that only explicit methods are used. Then use the M+1-k free parameters to increase stability and reduce truncation error.

In actual practice the best methods will be developed using a mixture of the features A - D.

5.6 NUMERICAL RESULTS

As an example of the optimization procedure, we have analyzed a conventional method with k=2 and order 3. Equation (5-1) becomes

$$y_m=a_1y_{m-1}+a_2y_{m-2}+h\ (b_oy'_m+b_1y'_{m-1}+b_2y_{m-2}) \quad (5\text{-}14)$$

We have optimized this method for solving the equations

$$y'=-y(x) \quad y(0)=10 \quad (5\text{-}15)$$

Table 5 provides the coefficients of the optimized method, while Table 6 compares the new method with the standard Adams-Moulton corrector. We used the same 3rd order predictor for both methods. As is indicated by the results considerable improvement was obtained by using the optimization procedures.

Table 5. Improved Method Corrector Coefficients, for k=2, Order is 3

	Predictor	Adams-Moulton	Optimized Method
a_1	-4	1	.001
a_2	5	0	.999
b_o	0	5/12	.3334
b_1	4	2/3	1.333
b_2	2	-1/12	.3329

Table 6. Computation Results After 8 Steps for an Improved Method

Step Size	Adams-Moulton	Optimized	True Solution
.05	7.0468|95	7.046881	7.046881
.1	4.966|050	4.9658|61	4.965853
.2	2.46|8801	2.466|114	2.465970
.3	1.2|40947	1.225|229	1.224564
.35	.8|993794	.86|40661	.8629359

For another example we have taken a cyclic method with M=2, k=2, order 3, utilized the free parameter to make one of the correctors explicit. This method is the Adams-Moulton corrector followed by the predictor utilized in Table 6. Computational results are presented in Table 7.

Table 7. Computational Results for a Cyclic Method Containing One Explicit Method, M=2, k=2, Order is 3

Step Size of .05

N	Cyclic-Explicit	True Solution
3	9.04837|7	9.048374
4	8.6070|59	8.607080
5	8.1872|88	8.187308
6	7.787|980	7.788008
7	7.4081|57	7.408182
8	7.0468|43	7.406881

REFERENCES

1. Dahlquist, Germund, Convergence and Stability in the Numerical Integration of Ordinary Differential Equations, Math. Scand. 4 (1956), pp. 33-53.

2. Butcher, J. C., A Modified Multistep Method for the Numerical Integration of Ordinary Differential Equations, J.A.C.M. 12, 1 (Jan. 1965), pp. 124-135.

3. Gregg, W. B., and Stetter, H. J., Generalized Multistep Predictor-Corrector Methods, J.A.C.M. 11, 2 (April 1964), pp. 188-209.

4. Hansen, Eldon, Cyclic Composite Multistep Predictor-Corrector Methods, Proc. 24th Natl. Confer. A.C.M. (1969), pp. 135-139.

5. Gear, C. W., The Numerical Integration of Ordinary Differential Equations, Math. Comp., Vol. 21 (1967), pp. 146-156.

6. Dyer, James, Generalized Multistep Methods in Satellite Orbit Computation, J.A.C.M., Vol. 15, No. 4 (Oct. 1968), pp. 712-719.

7. Velez, C. E., Brodsky, G. P., GEOSTAR-1 A Geopotential and Station Position Recovery System, Goddard Space Flight Center, Greenbelt, Maryland, Preprint X553-69-544, Nov. 1969, pp. 213-215.

8. Henrici, Peter, Discrete Variable Methods in Ordinary Differential Equations, John Wiley & Sons, Inc., N.Y. (1962), 400 pp.

9. Donelson, J. III, and Hansen, E., Cyclic Composite Multistep Predictor-Corrector Methods, Submitted for Publication (1970), 40 pp.

10. Sammet, J. E., and Bond, E. R., Introduction to FORMAC: A Formula Manipulation Compiler, IEEE Trans. Elect. Computers, Vol. EC-13 (Aug. 1964), pp. 386-394.

11. Maury, J. L., Jr., and Brodsky, G. P., Cowell Type Numerical Integration as Applied to Satellite Orbit Computation, Goddard S.F.C., Greenbelt, Maryland, Preprint X-553-69-46, (Dec. 1969).

12. Hamming, R. W., Numerical Methods for Scientists and Engineers, McGraw-Hill Co., N.Y. (1962).

REFERENCES
(Continued)

13. Nigro, B. J., An Investigation of Optimally Stable Numerical Integration Methods with Application to Real Time Simulation, Simulation, Volume 13: Number 5, November 1969.

14. Karim, A.I.A., A Theorem for the Stability of General Predictor Corrector Methods for the Solution of Systems of Differential Equations, Journal of the Association for Computing Machinery, Vol. 15, No. 4, October 1968, pp. 706-711.

15. Rahme, H. S., A New Look at the Numerical Integration of Ordinary Differential Equations, Journal of the ACM, June 1969.

APPENDIX A

THE GENERATION OF COEFFICIENTS FOR OFF-GRID GENERALIZED METHODS--PROGRAMS GENCOA AND GENCOB

by

JAMES DYER

THE GENERATION OF COEFFICIENTS FOR OFF-GRID GENERALIZED METHODS--PROGRAMS GENCOA AND GENCOB

A 1. INTRODUCTION

The coefficients entering into difference equations which constitute off-grid generalized methods are computed in a manner which is an extension of the method by which Butcher showed the existence of many new, stable, first-order methods [1]. In these programs we compute coefficients for first-order (Class I) methods by means of GENCOA and for special second-order (Class II) methods by means of GENCOB. The latter applies to the equation $y'' = f(x,y)$. High-order, stable, exllicit, and implicit methods of both classes are determined. Representatives fit together to form the generalized algorithms high-order analoges to the Gauss-Jackson algorithms. Butcher based his analysis on Hermite's formula [2] for interpolating a function and its first derivative on a set of $k + 1$ equally spaced points on the real line. We have generated coefficients with the aid of formulae which generalize Hermite's formula. The generalization consists in interpolating nonsequential derivatives; e.g., a condition on the second derivative of an interpolating polynomial may be imposed without a requirement on the first derivative. In this way, Class II difference equations may be constructed in analogy with the Adams second-order method or Cowell method.

The kind of generalized Hermite interpolation used in these programs is herein called "quasi-Hermite." In quasi-Hermite interpolating formulae, the

coefficients (quasi-Hermite interpolating functions of the proper kind) are not easily written by means of analytical formulae and are computed in determinantal form.

As will be seen, the program permits imposition of different derivative conditions at different points of the interpolating set. Thus, many different interpolating polynomials and methods are possible for each value of k.

The theory and construction of new, high-order numerical integration methods for second-order equations are described in some detail in [3].

A 2. MATHEMATICAL FORMULATION

To make use of the programs GENCOA and GENCOB, i.e., to understand the input and output, it is necessary to give a brief mathematical description of the function of these programs. The structure of the two programs is quite similar. For this reason the following discussion will pertain primarily to GENCOB used for Class II methods. The structure of GENCOA should be immediately apparent from the discussion. Differences between the two decks will be indicated whenever this is necessary for comprehension. The term "integration method" or simply "method" is used interchangeably with the term "difference equation." In our analysis, both are completely determined by an underlying osculatory interpolating polynomial. GENCOB is used to determine the coefficients in a difference equation for solving the differential equation

$$y'' = f(x,y) \qquad \text{(A-1)}$$

i.e., it is used to determine the coefficients α_i, β_i in the difference equation

$$\sum_{i=0}^{k} \alpha_i y_{n-i} + h^2 \sum_{i=0}^{k} \beta_i y''_{n-i} + h^2 \beta_\theta y''_{n-\theta} = 0 \qquad \text{(A-2)}$$

Here $y_i = y(x_i)$, $y''_i = f(x_i, y_i)$ and $\alpha_o = -1$. The upper value, k, of the indices is the step member of the difference equation, i.e., the number of units on the abscissa spanned by the information entering into it. The time interval of integration is h.

The coefficients α_i, β_i, are in turn computed from the quasi-Hermite form of an interpolating polynomial.

$$P(x) = \sum_{i=0}^{k} h_i(x) y_{n-i} + h^2 \sum_{i=0}^{k} \bar{h}_i(x) y''_{n-i} \qquad \text{(A-3)}$$

Here we suppose an interpolating set of points $S_k = \{x_n, x_{n-1}, \text{---} x_{n-k}\}$.

The functions (polynomials) h_i, $\bar{h}_i$ are the Hermitian interpolating functions of the first and second kind, respectively, mentioned in the output. (In the Class II case they use, in general, non-Hermitian functions.) As a control the output contains a printout of some matrices. It is therefore necessary to explain the meaning of these internally generated matrices.

If the conditions imposed to determine the h_i and $\bar{h}_i$ are Hermite (sequential), then the h_i and $\bar{h}_i$ have simple, well-known expressions. In the non-Hermite case and if the set of conditions changes, it is convenient to compute the h_i and $\bar{h}_i$ from their determinantal form. They appear as scaled subdeterminants of a matrix to be defined.

Assume k to be given. Find the polynomial P(x) of order 2k + 1 interpolating y(x) and its second derivative at all points of S_k. This polynomial can be written in the following manner. The subscript n is omitted from points X_{n-j} because the polynomial is essentially independent of it.

$$F(k,B,x) = \begin{vmatrix} P_{(x)} & 1 & x & x^2 & \cdots & x^{2k+1} \\ y_o & 1 & x_o & x_o^2 & \cdots & x_o^{2k+1} \\ y_1 & 1 & x_1 & x_1^2 & \cdots & x_1^{2k+1} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ y_k & 1 & x_k & x_k^2 & \cdots & x_k^{2k+1} \\ y_o'' & 0 & 0 & 2 & \cdots & (2k+1)2k\, x_o^{2k-1} \\ y_1'' & 0 & 0 & 2 & \cdots & (2k+1)2k\, x_1^{2k-1} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ y_k'' & 0 & 0 & 2 & \cdots & (2k+1)2k\, x_k^{2k-1} \end{vmatrix} \qquad (A\text{-}4)$$

The above form is different for GENCOA and GENCOB. This distinction is noted by writing F(k,B,x). The question of the uniqueness of such an interpolating polynomial P is considered in [3] and [4] and will not be treated here. Uniqueness depends on the conditions imposed and on the step number. Assuming uniqueness, it is clear by substitution and by differentiation and substitution that

$$F(k,B,x) \equiv 0$$

Expansion of the determinant by entries in the first column gives

$$DP(x) = \sum_{i=0}^{k} H_i(x)\, y_i + \sum_{i=0}^{k} \overline{H}_i(x)\, y_i'' \qquad \text{(A-5)}$$

It is clear how determinant D and polynomials H (x) and $\overline{H}$ (i) are defined. The determinant D is called the determinant of the fundamental matrix or the fundamental determinant of the general form F(k,B,x). D is nonzero if and only if the conditions imposed can be satisfied by at most one polynomial of degree 2k+1. When this is the case (A-5) can be divided by D to give the previous expressions. The H (x) and $\overline{H}$ (i) are then well defined.

The above general form (A-5) is meaningful if all possible conditions are imposed:

$$\left.\begin{aligned} P(x) &= y(x) \\ P''(x) &= y''(x) \end{aligned}\right\} \quad x \notin S_k.$$

Programs GENCOA and GENCOB provide for an arbitrary pattern of imposed function and derivative conditions. In the determinantal form of (A-4), there is one row for each function and derivative condition imposed. A similar, analogous form can be written, by elimination of certain rows, corresponding to certain eliminated conditions. An equal number of columns are eliminated. This is called the reduced determinantal form. To give this form a name, let τ and T be respectively the subsets of points of S_k at which function and derivative conditions are imposed. The reduced form depends on τ and T and can be written $F(k,B,\tau,T,hx)$. This is the canonical quasi-Hermite interpolating determinantal form associated with a set of imposed conditions. Its fundamental matrix is generated internally by GENCOB for coefficient determination.

References [3] and [4] may be consulted for a discussion of the question of uniqueness in the case of the reduced form. The situation is essentially the same as it was for the general form. If m+1 conditions are imposed, a polynomial of degree m is determined uniquely if and only if the fundamental matrix is nonsingular.

The dependence upon a particular interpolating set is unnecessary. The interpolating set S_k may be translated. Let us take $S_k = \{0, h, 2h, - - -, kh\}$. After some obvious manipulations permitted because the form F is identically equal to zero and after a change of variable, F(k,B,x) becomes;

$$F_c(k,B,hx) = \begin{vmatrix} P_{(hx)} & 1 & x & x^2 & & \cdots & x^{2k+1} \\ y_o & 1 & 0 & 0 & & \cdots & 0 \\ y_1 & 1 & 1 & 1 & & \cdots & 1 \\ \vdots & & & & & & \\ y_k & 1 & k & k^2 & - & \cdots & k^{2k+1} \\ h^2y''_0 & 0 & 0 & 2 & 0 & \cdots & 0 \\ h^2y''_1 & 0 & 0 & 2 & 6 & \cdots & (2k+1)2k \\ \vdots & & & & & & \\ h^2y''_k & 0 & 0 & 2 & 6 & \cdots & (2k+1)(2k)k^{2k-1} \end{vmatrix} \equiv 0 \qquad \text{(A-6)}$$

This is the canonical determinantal form associated with an interpolating problem in which a function condition and a derivative condition are imposed at each point.

If uniqueness holds, we may, as before, expand the reduced determinantal form, divide by the canonical fundamental determinant Δ and arrive at the quasi-Hermite interpolating polynomial.

$$P(hx) = \sum_{x_i \notin T} \frac{\Delta_i(x)}{\Delta} y_{n-i} + h^2 \sum_{x_i \notin T} \frac{\bar{\Delta}_i(x)}{\Delta} y''_{n-i} \qquad (A-7)$$

The degree of P is of course reduced by one for every condition eliminated.

Equation A-7 may be used as a general quasi-Hermite predictor for predicting grid or off-grid values. This equation can also be used as an off-grid corrector by differentiating twice. In this case it is assumed that $x_k \notin S_k$ and that $\Delta_k(x_{k-\theta h}) \neq 0$ so that the proper normalization can be done, i.e., the coefficient of y_k is -1. Comparing (A-7) with (A-2), it is apparent how the coefficients in the latter equation are defined and computed. Implicit and explicit methods may be derived by GENCOB. If the reduced form contains a row corresponding to the derivative condition at the rightmost point x_k, the resulting difference equation will in general be implicit. If not, the resulting difference equation will always be explicit. The coefficients for Class I difference equations are obtained by using the program GENCOA in essentially the same manner as has been described above. Differences in the determinantal forms produced by the two programs are obvious.

The concept of strong stability is paramount in this analysis. Therefore it is necessary to explain how the programs examine the stability of the difference equations produced.

The strong stability of a generalized off-grid difference equation (for zero step-size) of the type (A-2) is defined in the same manner as for the corresponding traditional equation [5], i.e., the equation is strongly stable if the roots of

$$\sum_{i=0}^{k} \alpha_i x^{k-i} = 0 \qquad \text{(A-8)}$$

are less than or equal to unity in magnitude. If a root has magnitude equal to unity, its multiplicity does not exceed one (Class I) or two (Class II). A consistent Class I equation has always a simple root of 1 and a Class II equation has a double root of 1. In practice, for an equation to be acceptable, it must have all other roots less than 1 in magnitude.

The coefficients α_i, $i \geq 1$ are functions of θ. The programs provide for the examination of the roots of stability equation (A-8) as θ takes on values of an arbitrarily fine mesh on an arbitrary interval on the real line. The manner of specification of the interval and the mesh will be described in the section on input.

It should be noted here that both programs GENCOA and GENCOB are capable of determining predicting equations which use an off-grid point, i.e., by inserting an input (TBAR) the program will internally impose a derivative condition at the point TBAR.

In generating the fundamental matrix of F_c (unreduced), the program will, if TBAR$\neq$0, replace the condition

$$P''(x_k) = y''(x_k)$$

by $P''(TBAR) = y''(TBAR)$

i.e., the last row of this matrix is changed. The rightmost function condition is then normally eliminated by correctly specifying the input parameters. How this is accomplished is described in the section on input.

A 3. SUBROUTINE DESCRIPTIONS

As stated earlier, the main routines are called GENCOA (for the generation of Class I coefficients) and GENCOB (Class II). Both programs have three subroutines: DET, SIGMA, and DNEWRA. A considerable portion of the computation is done within the main routines. A description of the Class I main routine GENCOA will be omitted because of its similarity to GENCOB. All are routines coded in FORTRAN IV in double precision for IBM 360 series computers.

A 3.1 Program GENCOB

This routine reads and prints input. The canonical k-step determinantal form $F_c(k,B,hx)$ (A-6) is computed.

The routine then computes the reduced canonical determinantal form $Fc(k,B,\tau,T,hx)$ and its fundamental determinant. From the reduced form the quasi-Hermite interpolating polynominal P(hx) is computed and normalized.

This polynomial and its second derivative are evaluated at the appropriate mesh points by calling subroutine SIGMA. Coefficients in the polynomials are determinants which are computed by calling subroutine DET. Also, from the second derivative, the stability equation (A-8) is formed and roots are computed by use of subroutine DNEWRA.

A 3.2 Subroutine DET

The determinant of a variable dimension matrix (generally, a submatrix of the reduced canonical determinantal form) is computed.

A 3.3 Subroutine SIGMA

The quasi-Hermite polynomial interpolating functions are evaluated and printed on the interval and at the mesh points specified by the input. The computation begins at the upper end of the interval and proceeds downward reducing the independent variable by the specified mesh norm. In addition, the derivatives of the above functions are evaluated and printed at the same points.

The α_i in equation (A-2) are essentially the set of the above mentioned derivatives associated with the function values. Polynomial equation (A-8) is formulated and the roots are found by calling subroutine DNEWRA. Roots are printed in the output.

A 3.4 Subroutine DNEWRA

This subroutine finds the roots of a polynomial of arbitrary degree. Possible errors are described in the section on usage. It is used for computation of the roots of (A-8).

A 4. PROGRAM USAGE

A 4.1 Input

For both programs for Class I and Class II; the number of input cards are the same. There are three input cards in most cases. They are as explained below:

(i) The first card contains the variables 'K, IP, BTHET, DTHET, ETHET, SCALE'. K and IP are integer variables and the other four are double-precision variables. The format in which the above card should be input is '2I5,4D10.3'.

(ii) The second card depends upon the value of IP in the first card. If the first card contains IP=0, the second card is not used. For values other than IP=0, the second card contains IØUT(I) and I goes from 1 to IP, and the format is '24I3'.

(iii) The third card contains TBAR, which is also explained in the description of the program. Its format is 'D10.3'.

A 4.2 Output

All the input parameters are printed out. The matrices A, B, and H and the fundamental determinant are all printed out.

Next, the variables L, TH, and T4 are printed out. The variable L is a counter; TH stands for the off-grid position THETA; and T4 stands for the off-grid coefficient β (BETA).

The Hermitian interpolating functions are printed out next. Below that the derivatives of the function are printed out. Next to that, the Hermitian interpolating functions of second kind are printed. Now the derivatives are printed. The Hermitian interpolating functions of the first kind and the second kind and their derivatives are explained in detail in the program description.

The roots of the stability polynomial are printed out. The roots are computed in the subroutine 'DNEWRA'. The roots of the polynomial are stored in the variable RR and all the values in the RR storage space are printed out in a sequence. The first two values printed correspond to one root of the polynomial. The first value is the real part of the root and the second value is the imaginary part.

All the information explained above, is printed out for every increment of THETA.

As noted before, the input as well as the output are the same for both Class I and Class II programs.

A 4.3 DEFINITIONS

K - The step-number of the method.

IP - The number of conditions to be eliminated in the determination of the underlying interpolating polynomial.

BTHET - Initial theta point.
DTHET - Terminal theta point
ETHET - Increment in theta

} These values specify the points at which the roots of (A-8) are computed.

SCALE - An arbitrary factor multiplying all entries of matrix A for convenience purposes.

IOUT(I) - Conditions to be eliminated.

TBAR - The point at which the derivative value is imposed (for use with off-grid predictors).

REFERENCES

1. J. C. Butcher. "A Modified Multistep Method for the Numerical Integration of Ordinary Differential Equations," Journal of the ACM, Vol. 12 (1965), pp 124-135.

2. Z. Kopal. Numerical Analysis, Second Edition, New York, John Wiley and Sons, 1961.

3. J. Dyer. "Generalized Multistep Methods in Satellite Orbit Computation," Journal of the ACM, Vol. 15 (1968), pp 712-719.

4. J. Dyer. "Quasi-Hermite Interpolation," System Development Corporation Document TM-3734 (December 1967).

5. P. Henrici. Discrete Variable Methods in Ordinary Differential Equations, New York, John Wiley and Sons, 1962.

APPENDIX B

TABLES AND COMPUTER PROGRAMS

FOR GENERATING CYCLIC METHODS

APPENDIX B (Cont'd.)

Table B 1. Parametric Solution of the Order Equations, Class I

Note: Some repeating decimals were truncated before 10 places even though they do repeat to 25 or more.

B 1-1.

K=3 Order 5	a_0	a_3
a_1	+0.7272727272	-1.7272727272
a_2	-1.7272727272	+0.7272727272
b_0	-0.3030303030	-0.0303030303
b_1	-1.7272727272	+0.7272727272
b_2	-0.7272727272	+1.7272727272
b_3	+0.0303030303	+0.3030303030

B 1-2.

K=4 Order 7	a_0	a_4
a_1	+3.9272727272	-2.4727272727
a_2	-2.4545454545	-2.4545454545
a_3	-2.4727272727	+3.9272727272
b_0	-0.2563636363	-0.0163636363
b_1	-3.0109090909	+0.8290909090
b_2	-4.3200000000	+4.3200000000
b_3	-0.8290909090	+3.0109090909
b_4	+0.0163636363	+0.2563636363

Table B 1. (Cont'd.)

B 1-3.

K=4 Order 8	a_4
a_0	-1.000000
a_1	-6.400000
a_2	+0.000000
a_3	+6.400000
b_0	+0.240000
b_1	+3.840000
b_2	+8.640000
b_3	+3.840000
b_4	+0.240000

Table B 2. Parametric Solution of the Order Equations, Class II

B 2-1.

K=3 Order 4	a_0	a_3
a_1	-2.00000000	+1.00000000
a_2	+1.00000000	-2.00000000
b_0	+0.08333333	+0.00000000
b_1	+0.83333333	+0.08333333
b_2	+0.08333333	+0.83333333
b_3	+0.00000000	+0.08333333

B 2-2.

K=3 Order 5	a_3
a_0	-1.00000000
a_1	3.00000000
a_2	-3.00000000
b_0	-0.08333333
b_1	-0.75000000
b_2	+0.75000000
b_3	+0.08333333

B 2-3.

K=4 Order 5	a_0	a_2	a_4
a_1	-1.50000000	-0.50000000	+0.50000000
a_3	+0.50000000	-0.50000000	-1.50000000
b_0	+0.0770833333	+0.0020833333	-0.0062500000
b_1	+0.90000000	-0.05000000	+0.06666666
b_2	+0.46250000	-0.40416666	+0.46250000
b_3	+0.06666666	-0.05000000	+0.90000000
b_4	-0.00625000	+0.00208333	+0.07708333

Table B 2. (Cont'd.)

B 2-4.

K=4 Order 6	a_2	a_4
a_0	+0.00000000	+1.00000000
a_1	-0.50000000	-1.00000000
a_3	-0.50000000	-1.00000000
b_0	+0.02083333	+0.07083333
b_1	-0.05000000	+0.96666666
b_2	-0.40416666	+0.92499999
b_3	-0.05000000	+0.96666666
b_4	+0.02083333	+0.07083333

B 2-5.

B=5 Order 8	a_0	a_5
a_1	+4.12903225806451612903225806	+1.0000
a_2	-10.25806451612903225806451 61	+4.12903225806451612903225806
a_3	+4.12903225806451612903225806	-10.25806451612903225806451 61
a_4	+1.0000	+4.12903225806451612903225806
b_0	+0.04946236559139784946236 5	+0.0000
b_1	+1.47956989247311827956989 24	+0.04946236559139784946236 5
b_2	+5.07096774193548387096774 19	+1.47956989241311827956989 24
b_3	+1.47956989247311827956989 2	+5.07096774193548387096774 19
b_4	+0.04946236559139784946236 5	+1.47956989247311827956989 2
b_5	+0.0000	+0.04946236559139784946236 5

Table B 3. Cyclic Method Coefficients, Class I

B 3-1.

m=	1	2	3	4
K=3, P=5	Order 5	Order 5	Order 5	
a_0^m	0	+1.50416666	The same as	
a_1^m	-1.727272	-0.63333333	for m=1	
a_2^m	+0.727272	-1.87083333		
a_3^m	+1	+1		
b_0^m	-0.030303	-0.48611111		
b_1^m	+0.727272	-1.87083333		
b_2^m	+1.727272	+0.63333333		
b_3^m	+0.303030	+0.34861111		

B 3-2.

K=4, P=7	Order 7	Order 7	Order 7	Order 7
a_0^m	0	+2.81324686097390074429844 83	Order 7	+0.84260398159537150927064 82
a_1^m	-2.47272727	+8.57566039946113746851754 23		+0.83640836408364083640836 40
a_2^m	-2.45454545	-9.35978774966321091782346 39		-4.52275522755227552275522
a_3^m	+3.92727272	-3.02911951077182729499252 66		+1.84374288187326317707621 52
a_4^m	+1	+1		+1
b_0^m	-0.01636363	-0.73757783163149091808378 401	The same as for m=1	-0.23237665709990433237665 710
b_1^m	+0.82909090	-7.64133963958687205919678 24		-1.70791307913079130791307 91
b_2^m	+4.32000000	-7.83322643940725121536929 65		+0.67995079950799507995079 95
b_3^m	+3.01090909	+0.67847169344709320109074 107		+2.31231378980456471231378 98
b_4^m	+0.25636363	+0.30023985849977547394521 564		+0.27015170151701517015170 15

Table B 3. (Cont'd.)

B 3-3.

m=	1	2	3	4
K=4, P=7 1/4	Order 8	Order 7	Order 7	Order 7
a_0^m	-1	+2.45925116793464102137785 13	0	+0.63499282590287289446241 47
a_1^m	-6.4	+7.18542276861604473850210 69	-2.47272727	+0.02106273445491900370693786
a_2^m	0	-8.49088923038502796156381 68	-2.45454545	-4.01316420903432437731683 62
a_3^m	+6.4	-2.15378470616565779831614 14	+3.92727272	+2.35710864867653247914748 36
a_4^m	+1	+1	+1	+1
b_0^m	+0.24	-0.64682620850688069820777 64	-0.01636363	-0.17915270627691832385309178
b_1^m	+3.84	-6.57549078927230096618494 86	+0.82909090	-1.08281476308210456950865 24
b_2^m	+8.64	-6.30396504547764921235231 76	+4.32000000	+1.57683099209958909592236 84
b_3^m	+3.84	+0.97196630440327944409399 96	+3.01090909	+2.48444231161507265477297 98
b_4^m	+0.24	+0.29660592820256685307709 21	+0.25636363	+0.26675442806022882918211 22

Table B 4. Cyclic Method Coefficients, Class II

B 4-1.

m=	1	2	3	4
K=3, P=4 2/3	Order 5	Order 5	Order 4	
a_0^m	-1	The same as for m=1	0	
a_1^m	+3		+1	
a_2^m	-3		-2	
a_3^m	+1		+1	
b_0^m	-1/12		0	
b_1^m	-3/4		+1/12	
b_2^m	+3/4		+10/12	
b_3^m	+1/12		+1/12	

B 4-2.

K=4, P=6	Order 6	Order 6	Order 6	Order 6
a_0^m	+1	+1	The same as for m=1	The same as for m=1
a_1^m	0	-1		
a_2^m	-2	0		
a_3^m	0	-1		
a_4^m	+1	+1		
b_0^m	+0.066666	+0.07083333		
b_1^m	+1.066666	+0.96666666		
b_2^m	+1.733333	+0.92500000		
b_3^m	+1.066666	+0.96666666		
b_4^m	+0.066666	+0.07083333		

Table B 4. (Cont'd.)

B 4-3.

m=	1	2	3	4
K=4 P=5 3/4	Order 6	Order 5	Order 6	Order 6
a_0^m	+1	0	The same as for m=1	The same as for m=1
a_1^m	-1	0		
a_2^m	0	+1		
a_3^m	-1	-2		
a_4^m	+1	+1		
b_0^m	+0.07083333	-0.00416666		
b_1^m	+0.96666666	+0.01666666		
b_2^m	+0.92500000	+0.05833333		
b_3^m	+0.96666666	+0.85000000		
b_4^m	+0.07083333	+0.07916666		

Table B 5. Coefficients of the Stability Polynomial

B 5-1 K=4,COEFFS OF THE CHARACTERISTIC (STABILITY) POLY

```
   50 F0   =        + A(1,5)  * A(2,5)  * A(3,5)  * A(4,5)
C
     F(1) =        + A(1,5)  * A(2,5)  * A(3,5)  * A(4,4)
     1             - A(1,5)  * A(2,5)  * A(3,1)  * A(4,3)
     2             + A(1,5)  * A(2,5)  * A(3,4)  * A(4,5)
     3             + A(1,5)  * A(2,1)  * A(3,1)  * A(4,2)
     4             - A(1,5)  * A(2,1)  * A(3,3)  * A(4,5)
     5             - A(1,5)  * A(2,2)  * A(3,5)  * A(4,2)
     6             + A(1,5)  * A(2,4)  * A(3,5)  * A(4,5)
     7             - A(1,1)  * A(2,1)  * A(3,1)  * A(4,1)
     8             + A(1,1)  * A(2,1)  * A(3,2)  * A(4,5)
     F(1) = F(1)   + A(1,1)  * A(2,2)  * A(3,5)  * A(4,1)
     1             - A(1,1)  * A(2,3)  * A(3,5)  * A(4,5)
     2             + A(1,2)  * A(2,5)  * A(3,1)  * A(4,1)
     3             - A(1,2)  * A(2,5)  * A(3,2)  * A(4,5)
     4             - A(1,3)  * A(2,5)  * A(3,5)  * A(4,1)
     5             + A(1,4)  * A(2,5)  * A(3,5)  * A(4,5)
C
     F(2) =        + A(1,5)  * A(2,5)  * A(3,4)  * A(4,4)
     1             - A(1,5)  * A(2,1)  * A(3,3)  * A(4,4)
     2             + A(1,5)  * A(2,2)  * A(3,3)  * A(4,3)
     3             - A(1,5)  * A(2,2)  * A(3,4)  * A(4,2)
     4             + A(1,5)  * A(2,4)  * A(3,5)  * A(4,4)
     5             - A(1,5)  * A(2,4)  * A(3,1)  * A(4,3)
     6             + A(1,5)  * A(2,4)  * A(3,4)  * A(4,5)
     7             + A(1,1)  * A(2,1)  * A(3,2)  * A(4,4)
     8             - A(1,1)  * A(2,2)  * A(3,2)  * A(4,3)
     F(2) = F(2)   + A(1,1)  * A(2,2)  * A(3,4)  * A(4,1)
     1             - A(1,1)  * A(2,3)  * A(3,5)  * A(4,4)
     2             + A(1,1)  * A(2,3)  * A(3,1)  * A(4,3)
     3             - A(1,1)  * A(2,3)  * A(3,4)  * A(4,5)
     4             - A(1,2)  * A(2,5)  * A(3,2)  * A(4,4)
     5             + A(1,2)  * A(2,2)  * A(3,2)  * A(4,2)
     6             - A(1,2)  * A(2,2)  * A(3,3)  * A(4,1)
     7             - A(1,2)  * A(2,3)  * A(3,1)  * A(4,2)
     8             + A(1,2)  * A(2,3)  * A(3,3)  * A(4,5)
     F(2) = F(2)   + A(1,2)  * A(2,4)  * A(3,1)  * A(4,1)
     1             - A(1,2)  * A(2,4)  * A(3,2)  * A(4,5)
     2             + A(1,3)  * A(2,5)  * A(3,2)  * A(4,3)
     3             - A(1,3)  * A(2,5)  * A(3,4)  * A(4,1)
     4             - A(1,3)  * A(2,1)  * A(3,2)  * A(4,2)
     5             + A(1,3)  * A(2,1)  * A(3,3)  * A(4,1)
     6             + A(1,3)  * A(2,3)  * A(3,5)  * A(4,2)
     7             - A(1,3)  * A(2,4)  * A(3,5)  * A(4,1)
     8             + A(1,4)  * A(2,5)  * A(3,5)  * A(4,4)
```

```
      F(2) = F(2) - A(1,4)  * A(2,5)  * A(3,1)  * A(4,3)
     1            + A(1,4)  * A(2,5)  * A(3,4)  * A(4,5)
     2            + A(1,4)  * A(2,1)  * A(3,1)  * A(4,2)
     3            - A(1,4)  * A(2,1)  * A(3,3)  * A(4,5)
     4            - A(1,4)  * A(2,2)  * A(3,5)  * A(4,2)
     5            + A(1,4)  * A(2,4)  * A(3,5)  * A(4,5)
C
      F(3) =       + A(1,5)  * A(2,4)  * A(3,4)  * A(4,4)
     1             - A(1,1)  * A(2,3)  * A(3,4)  * A(4,4)
     2             + A(1,2)  * A(2,3)  * A(3,3)  * A(4,4)
     3             - A(1,2)  * A(2,4)  * A(3,2)  * A(4,4)
     4             - A(1,3)  * A(2,3)  * A(3,3)  * A(4,3)
     5             + A(1,3)  * A(2,3)  * A(3,4)  * A(4,2)
     6             + A(1,3)  * A(2,4)  * A(3,2)  * A(4,3)
     7             - A(1,3)  * A(2,4)  * A(3,4)  * A(4,1)
     8             + A(1,4)  * A(2,5)  * A(3,4)  * A(4,4)
      F(3) = F(3) - A(1,4)  * A(2,1)  * A(3,3)  * A(4,4)
     1            + A(1,4)  * A(2,2)  * A(3,3)  * A(4,3)
     2            - A(1,4)  * A(2,2)  * A(3,4)  * A(4,2)
     3            + A(1,4)  * A(2,4)  * A(3,5)  * A(4,4)
     4            - A(1,4)  * A(2,4)  * A(3,1)  * A(4,3)
     5            + A(1,4)  * A(2,4)  * A(3,4)  * A(4,5)
C
      F(4) =       + A(1,4)  * A(2,4)  * A(3,4)  * A(4,4)
C
      FAL = F0*AL**0+F(1)*AL**1+F(2)*AL**2+F(3)*AL**3+F(4)*AL**4
C
C     K=4,THE STABILITY MATRIX ITSELF
C
      AB(1,1) = A(1,4) * AL + A(1,5)
      AB(1,2) = A(1,1)
      AB(1,3) = A(1,2)
      AB(1,4) = A(1,3)
      AB(2,1) = A(2,3) * AL
      AB(2,2) = A(2,4) * AL + A(2,5)
      AB(2,3) = A(2,1)
      AB(2,4) = A(2,2)
      AB(3,1) = A(3,2) * AL
      AB(3,2) = A(3,3) * AL
      AB(3,3) = A(3,4) * AL + A(3,5)
      AB(3,4) = A(3,1)
      AB(4,1) = A(4,1) * AL
      AB(4,2) = A(4,2) * AL
      AB(4,3) = A(4,3) * AL
      AB(4,4) = A(4,4) * AL + A(4,5)
C
```

B 5-2 K=5,COEFFS OF THE CHARACTERISTIC (STABILITY) POLY

```
   50 F0 =          + A(1,6)  * A(2,6)  * A(3,6)  * A(4,6)  * A(5,6)
C
    F(1) =          + A(1,5)  * A(2,6)  * A(3,6)  * A(4,6)  * A(5,6)
   1                + A(1,6)  * A(2,5)  * A(3,6)  * A(4,6)  * A(5,6)
   2                + A(1,6)  * A(2,6)  * A(3,5)  * A(4,6)  * A(5,6)
   3                + A(1,6)  * A(2,6)  * A(3,6)  * A(4,5)  * A(5,6)
   4                + A(1,6)  * A(2,6)  * A(3,6)  * A(4,6)  * A(5,5)
   5                - A(1,6)  * A(2,6)  * A(3,6)  * A(4,1)  * A(5,4)
   6                - A(1,6)  * A(2,6)  * A(3,1)  * A(4,4)  * A(5,6)
   7                + A(1,6)  * A(2,6)  * A(3,1)  * A(4,1)  * A(5,3)
   8                - A(1,6)  * A(2,6)  * A(3,2)  * A(4,6)  * A(5,3)
    F(1) = F(1)     - A(1,6)  * A(2,1)  * A(3,4)  * A(4,6)  * A(5,6)
   1                + A(1,6)  * A(2,1)  * A(3,1)  * A(4,3)  * A(5,6)
   2                - A(1,5)  * A(2,2)  * A(3,6)  * A(4,3)  * A(5,6)
   3                - A(1,6)  * A(2,1)  * A(3,1)  * A(4,1)  * A(5,2)
   4                + A(1,6)  * A(2,1)  * A(3,2)  * A(4,6)  * A(5,2)
   5                + A(1,6)  * A(2,2)  * A(3,6)  * A(4,1)  * A(5,2)
   6                - A(1,6)  * A(2,3)  * A(3,6)  * A(4,6)  * A(5,2)
   7                - A(1,1)  * A(2,4)  * A(3,6)  * A(4,6)  * A(5,6)
   8                + A(1,1)  * A(2,1)  * A(3,3)  * A(4,6)  * A(5,6)
    F(1) = F(1)     - A(1,2)  * A(2,6)  * A(3,3)  * A(4,6)  * A(5,6)
   1                - A(1,1)  * A(2,1)  * A(3,1)  * A(4,2)  * A(5,6)
   2                + A(1,1)  * A(2,2)  * A(3,6)  * A(4,2)  * A(5,6)
   3                + A(1,2)  * A(2,6)  * A(3,1)  * A(4,2)  * A(5,6)
   4                - A(1,3)  * A(2,6)  * A(3,6)  * A(4,2)  * A(5,6)
   5                + A(1,1)  * A(2,1)  * A(3,1)  * A(4,1)  * A(5,1)
   6                - A(1,1)  * A(2,1)  * A(3,2)  * A(4,6)  * A(5,1)
   7                - A(1,1)  * A(2,2)  * A(3,6)  * A(4,1)  * A(5,1)
   8                + A(1,1)  * A(2,3)  * A(3,6)  * A(4,6)  * A(5,1)
    F(1) = F(1)     - A(1,2)  * A(2,6)  * A(3,1)  * A(4,1)  * A(5,1)
   1                + A(1,2)  * A(2,6)  * A(3,2)  * A(4,6)  * A(5,1)
   2                + A(1,3)  * A(2,6)  * A(3,6)  * A(4,1)  * A(5,1)
   3                - A(1,4)  * A(2,6)  * A(3,6)  * A(4,6)  * A(5,1)
C
    F(2) =          + A(1,5)  * A(2,5)  * A(3,6)  * A(4,6)  * A(5,6)
   1                + A(1,5)  * A(2,6)  * A(3,5)  * A(4,6)  * A(5,6)
   2                + A(1,5)  * A(2,6)  * A(3,6)  * A(4,5)  * A(5,6)
   3                + A(1,5)  * A(2,6)  * A(3,6)  * A(4,6)  * A(5,5)
   4                - A(1,5)  * A(2,6)  * A(3,6)  * A(4,1)  * A(5,4)
   5                - A(1,5)  * A(2,6)  * A(3,1)  * A(4,4)  * A(5,6)
   6                + A(1,5)  * A(2,6)  * A(3,1)  * A(4,1)  * A(5,3)
   7                - A(1,5)  * A(2,6)  * A(3,2)  * A(4,6)  * A(5,3)
   8                - A(1,5)  * A(2,1)  * A(3,4)  * A(4,6)  * A(5,6)
    F(2) = F(2)     + A(1,5)  * A(2,1)  * A(3,1)  * A(4,3)  * A(5,6)
   1                - A(1,5)  * A(2,2)  * A(3,6)  * A(4,3)  * A(5,6)
   2                - A(1,5)  * A(2,1)  * A(3,1)  * A(4,1)  * A(5,2)
```

```
3                + A(1,5) *  A(2,1) *  A(3,2) *  A(4,6) *  A(5,2)
4                + A(1,5) *  A(2,2) *  A(3,6) *  A(4,1) *  A(5,2)
5                - A(1,5) *  A(2,3) *  A(3,6) *  A(4,6) *  A(5,2)
6                + A(1,6) *  A(2,5) *  A(3,5) *  A(4,6) *  A(5,6)
7                + A(1,6) *  A(2,5) *  A(3,6) *  A(4,5) *  A(5,6)
8                + A(1,6) *  A(2,5) *  A(3,6) *  A(4,6) *  A(5,5)
 F(2) = F(2)     + A(1,6) *  A(2,6) *  A(3,5) *  A(4,5) *  A(5,6)
1                + A(1,6) *  A(2,6) *  A(3,5) *  A(4,6) *  A(5,5)
2                + A(1,6) *  A(2,6) *  A(3,6) *  A(4,5) *  A(5,5)
3                - A(1,6) *  A(2,5) *  A(3,6) *  A(4,1) *  A(5,4)
4                - A(1,6) *  A(2,6) *  A(3,5) *  A(4,1) *  A(5,4)
5                - A(1,6) *  A(2,5) *  A(3,1) *  A(4,4) *  A(5,6)
6                - A(1,6) *  A(2,6) *  A(3,1) *  A(4,4) *  A(5,5)
7                + A(1,6) *  A(2,5) *  A(3,1) *  A(4,1) *  A(5,3)
8                + A(1,6) *  A(2,6) *  A(3,2) *  A(4,4) *  A(5,4)
 F(2) = F(2)     - A(1,6) *  A(2,5) *  A(3,2) *  A(4,6) *  A(5,3)
1                - A(1,6) *  A(2,6) *  A(3,2) *  A(4,5) *  A(5,3)
2                - A(1,6) *  A(2,1) *  A(3,4) *  A(4,6) *  A(5,5)
3                - A(1,6) *  A(2,1) *  A(3,4) *  A(4,5) *  A(5,6)
4                + A(1,6) *  A(2,1) *  A(3,4) *  A(4,1) *  A(5,4)
5                + A(1,6) *  A(2,2) *  A(3,4) *  A(4,4) *  A(5,6)
6                - A(1,6) *  A(2,2) *  A(3,4) *  A(4,1) *  A(5,3)
7                + A(1,6) *  A(2,3) *  A(3,4) *  A(4,6) *  A(5,3)
8                + A(1,6) *  A(2,1) *  A(3,1) *  A(4,3) *  A(5,5)
 F(2) = F(2)     - A(1,6) *  A(2,1) *  A(3,2) *  A(4,3) *  A(5,4)
1                - A(1,6) *  A(2,2) *  A(3,6) *  A(4,3) *  A(5,5)
2                - A(1,6) *  A(2,2) *  A(3,5) *  A(4,3) *  A(5,6)
3                + A(1,6) *  A(2,2) *  A(3,2) *  A(4,3) *  A(5,3)
4                + A(1,6) *  A(2,3) *  A(3,6) *  A(4,3) *  A(5,4)
5                - A(1,6) *  A(2,3) *  A(3,1) *  A(4,3) *  A(5,3)
6                + A(1,6) *  A(2,1) *  A(3,2) *  A(4,5) *  A(5,2)
7                + A(1,6) *  A(2,2) *  A(3,5) *  A(4,1) *  A(5,2)
8                - A(1,6) *  A(2,2) *  A(3,2) *  A(4,4) *  A(5,2)
 F(2) = F(2)     - A(1,6) *  A(2,3) *  A(3,6) *  A(4,5) *  A(5,2)
1                - A(1,6) *  A(2,3) *  A(3,5) *  A(4,6) *  A(5,2)
2                + A(1,6) *  A(2,3) *  A(3,1) *  A(4,4) *  A(5,2)
3                - A(1,1) *  A(2,4) *  A(3,5) *  A(4,6) *  A(5,6)
4                - A(1,1) *  A(2,4) *  A(3,6) *  A(4,5) *  A(5,6)
5                - A(1,1) *  A(2,4) *  A(3,6) *  A(4,6) *  A(5,5)
6                + A(1,1) *  A(2,4) *  A(3,6) *  A(4,1) *  A(5,4)
7                + A(1,1) *  A(2,4) *  A(3,1) *  A(4,4) *  A(5,6)
8                - A(1,1) *  A(2,4) *  A(3,1) *  A(4,1) *  A(5,3)
 F(2) = F(2)     + A(1,1) *  A(2,4) *  A(3,2) *  A(4,6) *  A(5,3)
1                + A(1,2) *  A(2,4) *  A(3,4) *  A(4,6) *  A(5,6)
2                - A(1,2) *  A(2,4) *  A(3,1) *  A(4,3) *  A(5,6)
3                + A(1,3) *  A(2,4) *  A(3,6) *  A(4,3) *  A(5,6)
4                + A(1,2) *  A(2,4) *  A(3,1) *  A(4,1) *  A(5,2)
```

```
5                - A(1,2) *  A(2,4) *  A(3,2) *  A(4,6) *  A(5,2)
6                - A(1,3) *  A(2,4) *  A(3,6) *  A(4,1) *  A(5,2)
7                + A(1,4) *  A(2,4) *  A(3,6) *  A(4,6) *  A(5,2)
8                + A(1,1) *  A(2,1) *  A(3,3) *  A(4,5) *  A(5,6)
 F(2) = F(2)     + A(1,1) *  A(2,1) *  A(3,3) *  A(4,6) *  A(5,5)
1                - A(1,1) *  A(2,1) *  A(3,3) *  A(4,1) *  A(5,4)
2                - A(1,1) *  A(2,2) *  A(3,3) *  A(4,4) *  A(5,6)
3                + A(1,1) *  A(2,2) *  A(3,3) *  A(4,1) *  A(5,3)
4                - A(1,1) *  A(2,3) *  A(3,3) *  A(4,6) *  A(5,3)
5                - A(1,2) *  A(2,5) *  A(3,3) *  A(4,6) *  A(5,6)
6                - A(1,2) *  A(2,6) *  A(3,3) *  A(4,5) *  A(5,6)
7                - A(1,2) *  A(2,6) *  A(3,3) *  A(4,6) *  A(5,5)
8                + A(1,2) *  A(2,6) *  A(3,3) *  A(4,1) *  A(5,4)
 F(2) = F(2)     + A(1,3) *  A(2,6) *  A(3,3) *  A(4,4) *  A(5,6)
1                - A(1,3) *  A(2,6) *  A(3,3) *  A(4,1) *  A(5,3)
2                + A(1,4) *  A(2,6) *  A(3,3) *  A(4,6) *  A(5,3)
3                + A(1,2) *  A(2,2) *  A(3,3) *  A(4,3) *  A(5,6)
4                - A(1,3) *  A(2,1) *  A(3,3) *  A(4,3) *  A(5,6)
5                - A(1,2) *  A(2,2) *  A(3,3) *  A(4,1) *  A(5,2)
6                + A(1,2) *  A(2,3) *  A(3,3) *  A(4,6) *  A(5,2)
7                + A(1,3) *  A(2,1) *  A(3,3) *  A(4,1) *  A(5,2)
8                - A(1,4) *  A(2,1) *  A(3,3) *  A(4,6) *  A(5,2)
 F(2) = F(2)     - A(1,1) *  A(2,1) *  A(3,1) *  A(4,2) *  A(5,5)
1                + A(1,1) *  A(2,1) *  A(3,2) *  A(4,2) *  A(5,4)
2                + A(1,1) *  A(2,2) *  A(3,5) *  A(4,2) *  A(5,6)
3                + A(1,1) *  A(2,2) *  A(3,6) *  A(4,2) *  A(5,5)
4                - A(1,1) *  A(2,2) *  A(3,2) *  A(4,2) *  A(5,3)
5                - A(1,1) *  A(2,3) *  A(3,6) *  A(4,2) *  A(5,4)
6                + A(1,1) *  A(2,3) *  A(3,1) *  A(4,2) *  A(5,3)
7                + A(1,2) *  A(2,5) *  A(3,1) *  A(4,2) *  A(5,6)
8                + A(1,2) *  A(2,6) *  A(3,1) *  A(4,2) *  A(5,5)
 F(2) = F(2)     - A(1,2) *  A(2,6) *  A(3,2) *  A(4,2) *  A(5,4)
1                - A(1,3) *  A(2,5) *  A(3,6) *  A(4,2) *  A(5,6)
2                - A(1,3) *  A(2,6) *  A(3,5) *  A(4,2) *  A(5,6)
3                - A(1,3) *  A(2,6) *  A(3,6) *  A(4,2) *  A(5,5)
4                + A(1,3) *  A(2,6) *  A(3,2) *  A(4,2) *  A(5,3)
5                + A(1,4) *  A(2,6) *  A(3,6) *  A(4,2) *  A(5,4)
6                - A(1,4) *  A(2,6) *  A(3,1) *  A(4,2) *  A(5,3)
7                - A(1,2) *  A(2,2) *  A(3,4) *  A(4,2) *  A(5,6)
8                + A(1,3) *  A(2,1) *  A(3,4) *  A(4,2) *  A(5,6)
 F(2) = F(2)     + A(1,2) *  A(2,2) *  A(3,2) *  A(4,2) *  A(5,2)
1                - A(1,2) *  A(2,3) *  A(3,1) *  A(4,2) *  A(5,2)
2                - A(1,3) *  A(2,1) *  A(3,2) *  A(4,2) *  A(5,2)
3                + A(1,3) *  A(2,3) *  A(3,6) *  A(4,2) *  A(5,2)
4                + A(1,4) *  A(2,1) *  A(3,1) *  A(4,2) *  A(5,2)
5                - A(1,4) *  A(2,2) *  A(3,6) *  A(4,2) *  A(5,2)
6                - A(1,1) *  A(2,1) *  A(3,2) *  A(4,5) *  A(5,1)
7                - A(1,1) *  A(2,2) *  A(3,5) *  A(4,1) *  A(5,1)
```

```
8                + A(1,1) *  A(2,2) *  A(3,2) *  A(4,4) *  A(5,1)
 F(2) = F(2)     + A(1,1) *  A(2,3) *  A(3,5) *  A(4,6) *  A(5,1)
1                + A(1,1) *  A(2,3) *  A(3,6) *  A(4,5) *  A(5,1)
2                - A(1,1) *  A(2,3) *  A(3,1) *  A(4,4) *  A(5,1)
3                - A(1,2) *  A(2,5) *  A(3,1) *  A(4,1) *  A(5,1)
4                + A(1,2) *  A(2,5) *  A(3,2) *  A(4,6) *  A(5,1)
5                + A(1,2) *  A(2,6) *  A(3,2) *  A(4,5) *  A(5,1)
6                + A(1,3) *  A(2,5) *  A(3,6) *  A(4,1) *  A(5,1)
7                + A(1,3) *  A(2,6) *  A(3,5) *  A(4,1) *  A(5,1)
8                - A(1,3) *  A(2,6) *  A(3,2) *  A(4,4) *  A(5,1)
 F(2) = F(2)     - A(1,4) *  A(2,5) *  A(3,6) *  A(4,6) *  A(5,1)
1                - A(1,4) *  A(2,6) *  A(3,5) *  A(4,6) *  A(5,1)
2                - A(1,4) *  A(2,6) *  A(3,6) *  A(4,5) *  A(5,1)
3                + A(1,4) *  A(2,6) *  A(3,1) *  A(4,4) *  A(5,1)
4                + A(1,2) *  A(2,2) *  A(3,4) *  A(4,1) *  A(5,1)
5                - A(1,2) *  A(2,3) *  A(3,4) *  A(4,6) *  A(5,1)
6                - A(1,3) *  A(2,1) *  A(3,4) *  A(4,1) *  A(5,1)
7                + A(1,4) *  A(2,1) *  A(3,4) *  A(4,6) *  A(5,1)
8                - A(1,2) *  A(2,2) *  A(3,2) *  A(4,3) *  A(5,1)
 F(2) = F(2)     + A(1,2) *  A(2,3) *  A(3,1) *  A(4,3) *  A(5,1)
1                + A(1,3) *  A(2,1) *  A(3,2) *  A(4,3) *  A(5,1)
2                - A(1,3) *  A(2,3) *  A(3,6) *  A(4,3) *  A(5,1)
3                - A(1,4) *  A(2,1) *  A(3,1) *  A(4,3) *  A(5,1)
4                + A(1,4) *  A(2,2) *  A(3,6) *  A(4,3) *  A(5,1)

 F(3) =          + A(1,6)  * A(2,5)  * A(3,5)  * A(4,5)  * A(5,6)
1                + A(1,6)  * A(2,5)  * A(3,5)  * A(4,6)  * A(5,5)
2                + A(1,6)  * A(2,5)  * A(3,6)  * A(4,5)  * A(5,5)
3                + A(1,6)  * A(2,6)  * A(3,5)  * A(4,5)  * A(5,5)
4                - A(1,6)  * A(2,5)  * A(3,5)  * A(4,1)  * A(5,4)
5                - A(1,6)  * A(2,5)  * A(3,1)  * A(4,4)  * A(5,5)
6                + A(1,6)  * A(2,5)  * A(3,2)  * A(4,4)  * A(5,4)
7                - A(1,6)  * A(2,5)  * A(3,2)  * A(4,5)  * A(5,3)
8                - A(1,6)  * A(2,1)  * A(3,4)  * A(4,5)  * A(5,5)
 F(3) = F(3)     + A(1,6)  * A(2,2)  * A(3,4)  * A(4,4)  * A(5,5)
1                - A(1,6)  * A(2,3)  * A(3,4)  * A(4,4)  * A(5,4)
2                + A(1,6)  * A(2,3)  * A(3,4)  * A(4,5)  * A(5,3)
3                - A(1,6)  * A(2,2)  * A(3,5)  * A(4,3)  * A(5,5)
4                + A(1,6)  * A(2,3)  * A(3,5)  * A(4,3)  * A(5,4)
5                - A(1,6)  * A(2,3)  * A(3,5)  * A(4,5)  * A(5,2)
6                - A(1,1)  * A(2,4)  * A(3,5)  * A(4,5)  * A(5,6)
7                - A(1,1)  * A(2,4)  * A(3,5)  * A(4,6)  * A(5,5)
8                - A(1,1)  * A(2,4)  * A(3,6)  * A(4,5)  * A(5,5)
 F(3) = F(3)     + A(1,1)  * A(2,4)  * A(3,5)  * A(4,1)  * A(5,4)
1                + A(1,1)  * A(2,4)  * A(3,1)  * A(4,4)  * A(5,5)
2                - A(1,1)  * A(2,4)  * A(3,2)  * A(4,4)  * A(5,4)
3                + A(1,1)  * A(2,4)  * A(3,2)  * A(4,5)  * A(5,3)
4                + A(1,2)  * A(2,4)  * A(3,4)  * A(4,5)  * A(5,6)
```

```
5              + A(1,2)  * A(2,4)  * A(3,4)  * A(4,6)  * A(5,5)
6              - A(1,2)  * A(2,4)  * A(3,4)  * A(4,1)  * A(5,4)
7              - A(1,3)  * A(2,4)  * A(3,4)  * A(4,4)  * A(5,6)
8              + A(1,3)  * A(2,4)  * A(3,4)  * A(4,1)  * A(5,3)
 F(3) = F(3)   - A(1,4)  * A(2,4)  * A(3,4)  * A(4,6)  * A(5,3)
1              - A(1,2)  * A(2,4)  * A(3,1)  * A(4,3)  * A(5,5)
2              + A(1,2)  * A(2,4)  * A(3,2)  * A(4,3)  * A(5,4)
3              + A(1,3)  * A(2,4)  * A(3,5)  * A(4,3)  * A(5,6)
4              + A(1,3)  * A(2,4)  * A(3,6)  * A(4,3)  * A(5,5)
5              - A(1,3)  * A(2,4)  * A(3,2)  * A(4,3)  * A(5,3)
6              - A(1,4)  * A(2,4)  * A(3,6)  * A(4,3)  * A(5,4)
7              + A(1,4)  * A(2,4)  * A(3,1)  * A(4,3)  * A(5,3)
8              - A(1,2)  * A(2,4)  * A(3,2)  * A(4,5)  * A(5,2)
 F(3) = F(3)   - A(1,3)  * A(2,4)  * A(3,5)  * A(4,1)  * A(5,2)
1              + A(1,3)  * A(2,4)  * A(3,2)  * A(4,4)  * A(5,2)
2              + A(1,4)  * A(2,4)  * A(3,5)  * A(4,6)  * A(5,2)
3              + A(1,4)  * A(2,4)  * A(3,6)  * A(4,5)  * A(5,2)
4              - A(1,4)  * A(2,4)  * A(3,1)  * A(4,4)  * A(5,2)
5              + A(1,1)  * A(2,1)  * A(3,3)  * A(4,5)  * A(5,5)
6              - A(1,1)  * A(2,2)  * A(3,3)  * A(4,4)  * A(5,5)
7              + A(1,1)  * A(2,3)  * A(3,3)  * A(4,4)  * A(5,4)
8              - A(1,1)  * A(2,3)  * A(3,3)  * A(4,5)  * A(5,3)
 F(3) = F(3)   - A(1,2)  * A(2,5)  * A(3,3)  * A(4,5)  * A(5,6)
1              - A(1,2)  * A(2,5)  * A(3,3)  * A(4,6)  * A(5,5)
2              - A(1,2)  * A(2,6)  * A(3,3)  * A(4,5)  * A(5,5)
3              + A(1,2)  * A(2,5)  * A(3,3)  * A(4,1)  * A(5,4)
4              + A(1,3)  * A(2,5)  * A(3,3)  * A(4,4)  * A(5,6)
5              + A(1,3)  * A(2,6)  * A(3,3)  * A(4,4)  * A(5,5)
6              - A(1,3)  * A(2,5)  * A(3,3)  * A(4,1)  * A(5,3)
7              - A(1,4)  * A(2,6)  * A(3,3)  * A(4,4)  * A(5,4)
8              + A(1,4)  * A(2,5)  * A(3,3)  * A(4,6)  * A(5,3)
 F(3) = F(3)   + A(1,4)  * A(2,6)  * A(3,3)  * A(4,5)  * A(5,3)
1              + A(1,2)  * A(2,2)  * A(3,3)  * A(4,3)  * A(5,5)
2              - A(1,2)  * A(2,3)  * A(3,3)  * A(4,3)  * A(5,4)
3              - A(1,3)  * A(2,1)  * A(3,3)  * A(4,3)  * A(5,5)
4              + A(1,3)  * A(2,3)  * A(3,3)  * A(4,3)  * A(5,3)
5              + A(1,4)  * A(2,1)  * A(3,3)  * A(4,3)  * A(5,4)
6              - A(1,4)  * A(2,2)  * A(3,3)  * A(4,3)  * A(5,3)
7              + A(1,2)  * A(2,3)  * A(3,3)  * A(4,5)  * A(5,2)
8              - A(1,3)  * A(2,3)  * A(3,3)  * A(4,4)  * A(5,2)
 F(3) = F(3)   - A(1,4)  * A(2,1)  * A(3,3)  * A(4,5)  * A(5,2)
1              + A(1,4)  * A(2,2)  * A(3,3)  * A(4,4)  * A(5,2)
2              + A(1,1)  * A(2,2)  * A(3,5)  * A(4,2)  * A(5,5)
3              - A(1,1)  * A(2,3)  * A(3,5)  * A(4,2)  * A(5,4)
4              + A(1,2)  * A(2,5)  * A(3,1)  * A(4,2)  * A(5,5)
5              - A(1,2)  * A(2,5)  * A(3,2)  * A(4,2)  * A(5,4)
6              - A(1,3)  * A(2,5)  * A(3,5)  * A(4,2)  * A(5,6)
7              - A(1,3)  * A(2,5)  * A(3,6)  * A(4,2)  * A(5,5)
```

```
8              - A(1,3)  * A(2,6)  * A(3,5)  * A(4,2)  * A(5,5)
 F(3) = F(3)   + A(1,3)  * A(2,5)  * A(3,2)  * A(4,2)  * A(5,3)
1              + A(1,4)  * A(2,5)  * A(3,6)  * A(4,2)  * A(5,4)
2              + A(1,4)  * A(2,6)  * A(3,5)  * A(4,2)  * A(5,4)
3              - A(1,4)  * A(2,5)  * A(3,1)  * A(4,2)  * A(5,3)
4              - A(1,2)  * A(2,2)  * A(3,4)  * A(4,2)  * A(5,5)
5              + A(1,2)  * A(2,3)  * A(3,4)  * A(4,2)  * A(5,4)
6              + A(1,3)  * A(2,1)  * A(3,4)  * A(4,2)  * A(5,5)
7              - A(1,3)  * A(2,3)  * A(3,4)  * A(4,2)  * A(5,3)
8              - A(1,4)  * A(2,1)  * A(3,4)  * A(4,2)  * A(5,4)
 F(3) = F(3)   + A(1,4)  * A(2,2)  * A(3,4)  * A(4,2)  * A(5,3)
1              + A(1,3)  * A(2,3)  * A(3,5)  * A(4,2)  * A(5,2)
2              - A(1,4)  * A(2,2)  * A(3,5)  * A(4,2)  * A(5,2)
3              + A(1,1)  * A(2,3)  * A(3,5)  * A(4,5)  * A(5,1)
4              + A(1,2)  * A(2,5)  * A(3,2)  * A(4,5)  * A(5,1)
5              + A(1,3)  * A(2,5)  * A(3,5)  * A(4,1)  * A(5,1)
6              - A(1,3)  * A(2,5)  * A(3,2)  * A(4,4)  * A(5,1)
7              - A(1,4)  * A(2,5)  * A(3,5)  * A(4,6)  * A(5,1)
8              - A(1,4)  * A(2,5)  * A(3,6)  * A(4,5)  * A(5,1)
 F(3) = F(3)   - A(1,4)  * A(2,6)  * A(3,5)  * A(4,5)  * A(5,1)
1              + A(1,4)  * A(2,5)  * A(3,1)  * A(4,4)  * A(5,1)
2              - A(1,2)  * A(2,3)  * A(3,4)  * A(4,5)  * A(5,1)
3              + A(1,3)  * A(2,3)  * A(3,4)  * A(4,4)  * A(5,1)
4              + A(1,4)  * A(2,1)  * A(3,4)  * A(4,5)  * A(5,1)
5              - A(1,4)  * A(2,2)  * A(3,4)  * A(4,4)  * A(5,1)
6              - A(1,3)  * A(2,3)  * A(3,5)  * A(4,3)  * A(5,1)
7              + A(1,4)  * A(2,2)  * A(3,5)  * A(4,3)  * A(5,1)
8              + A(1,5)  * A(2,5)  * A(3,5)  * A(4,6)  * A(5,6)
 F(3) = F(3)   + A(1,5)  * A(2,5)  * A(3,6)  * A(4,5)  * A(5,6)
1              + A(1,5)  * A(2,5)  * A(3,6)  * A(4,6)  * A(5,5)
2              + A(1,5)  * A(2,6)  * A(3,5)  * A(4,5)  * A(5,6)
3              + A(1,5)  * A(2,6)  * A(3,5)  * A(4,6)  * A(5,5)
4              + A(1,5)  * A(2,6)  * A(3,6)  * A(4,5)  * A(5,5)
5              - A(1,5)  * A(2,5)  * A(3,6)  * A(4,1)  * A(5,4)
6              - A(1,5)  * A(2,6)  * A(3,5)  * A(4,1)  * A(5,4)
7              - A(1,5)  * A(2,5)  * A(3,1)  * A(4,4)  * A(5,6)
8              - A(1,5)  * A(2,6)  * A(3,1)  * A(4,4)  * A(5,5)
 F(3) = F(3)   + A(1,5)  * A(2,5)  * A(3,1)  * A(4,1)  * A(5,3)
1              + A(1,5)  * A(2,6)  * A(3,2)  * A(4,4)  * A(5,4)
2              - A(1,5)  * A(2,5)  * A(3,2)  * A(4,6)  * A(5,3)
3              - A(1,5)  * A(2,6)  * A(3,2)  * A(4,5)  * A(5,3)
4              - A(1,5)  * A(2,1)  * A(3,4)  * A(4,6)  * A(5,5)
5              - A(1,5)  * A(2,1)  * A(3,4)  * A(4,5)  * A(5,6)
6              + A(1,5)  * A(2,1)  * A(3,4)  * A(4,1)  * A(5,4)
7              + A(1,5)  * A(2,2)  * A(3,4)  * A(4,4)  * A(5,6)
8              - A(1,5)  * A(2,2)  * A(3,4)  * A(4,1)  * A(5,3)
 F(3) = F(3)   + A(1,5)  * A(2,3)  * A(3,4)  * A(4,6)  * A(5,3)
1              + A(1,5)  * A(2,1)  * A(3,1)  * A(4,3)  * A(5,5)
```

```
      2              - A(1,5)  * A(2,1)  * A(3,2)  * A(4,3)  * A(5,4)
      3              - A(1,5)  * A(2,2)  * A(3,6)  * A(4,3)  * A(5,5)
      4              - A(1,5)  * A(2,2)  * A(3,5)  * A(4,3)  * A(5,6)
      5              + A(1,5)  * A(2,2)  * A(3,2)  * A(4,3)  * A(5,3)
      6              + A(1,5)  * A(2,3)  * A(3,6)  * A(4,3)  * A(5,4)
      7              - A(1,5)  * A(2,3)  * A(3,1)  * A(4,3)  * A(5,3)
      8              + A(1,5)  * A(2,1)  * A(3,2)  * A(4,5)  * A(5,2)
       F(3) = F(3)   + A(1,5)  * A(2,2)  * A(3,5)  * A(4,1)  * A(5,2)
      1              - A(1,5)  * A(2,2)  * A(3,2)  * A(4,4)  * A(5,2)
      2              - A(1,5)  * A(2,3)  * A(3,6)  * A(4,5)  * A(5,2)
      3              - A(1,5)  * A(2,3)  * A(3,5)  * A(4,6)  * A(5,2)
      4              + A(1,5)  * A(2,3)  * A(3,1)  * A(4,4)  * A(5,2)
C
       F(4) =        + A(1,5)  * A(2,5)  * A(3,5)  * A(4,5)  * A(5,6)
      1              + A(1,5)  * A(2,5)  * A(3,5)  * A(4,6)  * A(5,5)
      2              + A(1,5)  * A(2,5)  * A(3,6)  * A(4,5)  * A(5,5)
      3              + A(1,5)  * A(2,6)  * A(3,5)  * A(4,5)  * A(5,5)
      4              - A(1,5)  * A(2,5)  * A(3,5)  * A(4,1)  * A(5,4)
      5              - A(1,5)  * A(2,5)  * A(3,1)  * A(4,4)  * A(5,5)
      6              + A(1,5)  * A(2,5)  * A(3,2)  * A(4,4)  * A(5,4)
      7              - A(1,5)  * A(2,5)  * A(3,2)  * A(4,5)  * A(5,3)
      8              - A(1,5)  * A(2,1)  * A(3,4)  * A(4,5)  * A(5,5)
       F(4) = F(4)   + A(1,5)  * A(2,2)  * A(3,4)  * A(4,4)  * A(5,5)
      1              - A(1,5)  * A(2,3)  * A(3,4)  * A(4,4)  * A(5,4)
      2              + A(1,5)  * A(2,3)  * A(3,4)  * A(4,5)  * A(5,3)
      3              - A(1,5)  * A(2,2)  * A(3,5)  * A(4,3)  * A(5,5)
      4              + A(1,5)  * A(2,3)  * A(3,5)  * A(4,3)  * A(5,4)
      5              - A(1,5)  * A(2,3)  * A(3,5)  * A(4,5)  * A(5,2)
      6              - A(1,1)  * A(2,4)  * A(3,5)  * A(4,5)  * A(5,5)
      7              + A(1,2)  * A(2,4)  * A(3,4)  * A(4,5)  * A(5,5)
      8              - A(1,3)  * A(2,4)  * A(3,4)  * A(4,4)  * A(5,5)
       F(4) = F(4)   + A(1,4)  * A(2,4)  * A(3,4)  * A(4,4)  * A(5,4)
      1              - A(1,4)  * A(2,4)  * A(3,4)  * A(4,5)  * A(5,3)
      2              + A(1,3)  * A(2,4)  * A(3,5)  * A(4,3)  * A(5,5)
      3              - A(1,4)  * A(2,4)  * A(3,5)  * A(4,3)  * A(5,4)
      4              + A(1,4)  * A(2,4)  * A(3,5)  * A(4,5)  * A(5,2)
      5              - A(1,2)  * A(2,5)  * A(3,3)  * A(4,5)  * A(5,5)
      6              + A(1,3)  * A(2,5)  * A(3,3)  * A(4,4)  * A(5,5)
      7              - A(1,4)  * A(2,5)  * A(3,3)  * A(4,4)  * A(5,4)
      8              + A(1,4)  * A(2,5)  * A(3,3)  * A(4,5)  * A(5,3)
       F(4) = F(4)   - A(1,3)  * A(2,5)  * A(3,5)  * A(4,2)  * A(5,5)
      1              + A(1,4)  * A(2,5)  * A(3,5)  * A(4,2)  * A(5,4)
      2              - A(1,4)  * A(2,5)  * A(3,5)  * A(4,5)  * A(5,1)
      3              + A(1,6)  * A(2,5)  * A(3,5)  * A(4,5)  * A(5,5)
C
       F(5) =        + A(1,5)  * A(2,5)  * A(3,5)  * A(4,5)  * A(5,5)
C
```

```
      FAL  = F0*AL**0 + F(1)*AL**1 + F(2)*AL**2 + F(3)*AL**3
     C     + F(4)*AL**4 + F(5)*AL**5
C
C     K=5, THE STABILITY MATRIX ITSELF
C
      AB(1,1) = A(1,5) *AL + A(1,6)
      AB(1,2) = A(1,1)
      AB(1,3) = A(1,2)
      AB(1,4) = A(1,3)
      AB(1,5) = A(1,4)
      AB(2,1) = A(2,4) *AL
      AB(2,2) = A(2,5) *AL + A(2,6)
      AB(2,3) = A(2,1)
      AB(2,4) = A(2,2)
      AB(2,5) = A(2,3)
      AB(3,1) = A(3,3) *AL
      AB(3,2) = A(3,4) *AL
      AB(3,3) = A(3,5) *AL + A(3,6)
      AB(3,4) = A(3,1)
      AB(3,5) = A(3,2)
      AB(4,1) = A(4,2) *AL
      AB(4,2) = A(4,3) *AL
      AB(4,3) = A(4,4) *AL
      AB(4,4) = A(4,5) *AL + A(4,6)
      AB(4,5) = A(4,1)
      AB(5,1) = A(5,1) *AL
      AB(5,2) = A(5,2) *AL
      AB(5,3) = A(5,3) *AL
      AB(5,4) = A(5,4)      *AL
      AB(5,5) = A(5,5) *AL + A(5,6)
```

B 6. DESCRIPTION OF COMPUTER PROGRAMS FOR USE IN GENERATING CYCLIC METHODS

All programs were run on the CDC 6600 in Los Angeles, California. The language used is FORTRAN. All calculations were done in double precision carrying more than 27 significant digits.

B 6.1 STABIL

The first purpose of this program is to check the correctness of the expressions given for the coefficients, F_i, i = 0, ..., k, of the stability polynomial, $p(\lambda)$, (3-11) in terms of the coefficients a_i^m, m=1, ..., k; i = 0, ..., k. The second purpose is to map the values of the F_i when the solutions of the order equations are used.

These purposes are accomplished by evaluating the expressions for F(0), F(1), ..., F(k) in terms of the variables A(m,i), m = 1, ..., k; i = 1, ..., k + 1, where we have used the convention a_o^m = A(m,k+1). In the first case, sets of arbitrary values for all the A(m,i) are chosen. In the second, only the free A(m,i) are chosen arbitrarily, the others are determined from normalization and table (B 1) or (B 2). In either case, arbitrary values for λ, AL, are chosen and $p(\lambda)$ is computed as FAL = FO*AL**0 + F(1)*AL**1 + ... + F(K)*AL**K. Then the stability matrix, AB, (3-6) is written down in terms of AL and the A(m,i), and its determinant, DETERM is evaluated using Gaussian complete pivoting. The difference |FAL - DETERM| is computed. The F's, FAL, DETERM, and the above difference are printed out and an error message is printed out if this error is larger than a preassigned amount. For the expansions given in table B 5. the

relative error $|FAL - DETERM| / |DETERM|$ was always $< 10^{-25}$. The billable computing time was about 5 seconds for 100 sets of values.

B 6.2 ORDER

The purposes of this program are to solve the linear Class I and Class II order equations for arbitrary k, of arbitrary order, with an arbitrary number of parameters depending only on the input data given the program. Some results are given in tables B 1 and B 2.

These purposes are accomplished by writing the desired order equations in the form AX = BY where A is an NxN matrix, X is the N dimensional column vector of the coefficients a_i, b_i, i = 0, ..., k for which we wish to solve parametrically, B is an NxM dimensional matrix, Y is the M dimensional column vector of the remaining coefficients which we wish to leave as parameters, and N + M = 2k+2 = the total number of a's and b's. For example, for k = 4, Class I, order 7 we chose a_0 and a_4 as the free parameters, M = 2, N = 8, $Y = \begin{pmatrix} a0 \\ a_4 \end{pmatrix}$, B is the coefficients of a_0 and a_4, X is the remaining a's and b's, and A is their coefficients.

A and B are punched on cards and read by rows into the program. A is stored in ASTORE. X is computed using Gaussian elimination in double precision with complete pivoting (subroutine MATINV) and the coefficients of the first free parameter are stored in the first column of B, the second in the second, etc. First column of B new = A^{-1} x(first column of B old) etc. so the solution is of the form $X = A^{-1}BY$. A^{-1} and B (which now contains the parametric solution) are printed out. As an accuracy check ASTORE (containing A) is multiplied by B (containing the solution), the product is stored in BSTORE and printed out. This product should equal the original B. In all cases the maximum error of all the components was $\leq 10^{-24}$. The total billable computing time for one matrix was about 1 second.

B 6.3 NONLIN

The purposes of this program are to solve an arbitrary number of nonlinear stability equations (algorithms 1 and 2) for Class I and Class II depending on the nonlinear functions and which set of order equations is used. Some results are given by (3-15), (3-21), and (3-24).

These purposes are accomplished by using a modification of Newton's method in several dimensions known as the secant method. The great advantage of this method is that the partial derivatives of the nonlinear functions do not have to be computed, thus saving considerable manual labor. On a test case with a known solution Newton's method converged in four iterations and the secant in six (only slightly longer) to a relative error of 10^{-24} with approximately the same starting values.

First a maximum number of iterations and a maximum allowable relative error convergence criteria are established. Starting values are supplied on data cards. These are read and the iterations begin. There is a check to be sure we do not divide by zero in approximating the derivatives. In subroutine DELTAX, the nonlinear functions, F(I), I = 1, ..., N are evaluated at the starting points and the partial derivatives are approximated by partial difference quotients thus obtaining an approximate Jacobian, AJ, an N x N matrix. Subroutine FABX computes F(I) by substituting the starting values, normalization, and choices for the remaining free parameters into formulae representing table (B.1) or (B.2) and these values are then substituted into the proper F(I) given in table (B.5).

The next approximation to the solution is found by the vector equation XIP1 = XI - $(AJ)^{-1}F$ where $(AJ)^{-1}F$ is computed using subroutine MATINV. The relative error, RELERR = ||XIP1 - XI|| / ||XIP1||, is then calculated and compared with the allowable error as a test for convergence. Iteration is stopped when either convergence or the maximum number of iterations is obtained. The next starting values (if any) are then read and the process repeats. In most cases a relative error of 10^{-30} was obtained within 10 iterations using only a couple of seconds billable computer time.

B 6.4 PUTTOG

The purposes of this program are to put together the solutions of the linear and nonlinear equations, compute and punch the coefficients a_i^m, b_i^m and, compute values for the C_i (3-9) to be sure each single method is of the right order. Results are given in tables (B.3) and B.4).

This was done in the obvious way. The formulae for the C_i were obtained independently from those used in ORDER. The largest C_i which should have been zero was 10^{-27}. Total billable computing time for one cyclic method was less than 1 second.

The last step in checking out the coefficients was to use them to calculate the F_i (see STABIL) to be sure all roots were correct.

APPENDIX C

PROGRAMS FOR GENERATING COEFFICIENTS FOR MODIFIED MULTISTEP METHODS

PROGRAMS FOR GENERATING COEFFICIENTS FOR MODIFIED MULTISTEP METHODS

There are two programs written to compute modified multistep method coefficients--one for Class I and the other for Class II methods. Both programs have the same number of subroutines with the same subroutine names. The programs are written in double precision FORTRAN IV to be used on the CDC 6600 computer. The CDC 6600 computer gives about 28 decimal digits accuracy in double precision computation. This reduces round-off error considerably even for large numbers of operations. The programs are written in such a way that with only a few changes the programs can be run on the IBM 360 and the UNIVAC 1108 model computers.

C 1. COMPUTATION PROCEDURE

The programs implement the computation procedure and formulas described in 4.3 and 4.4. Specifically the following sequence is followed:

1. Generate Q_{NM} using either equation (4-14) or (4-15)
2. Generate Q_{NA} using either equation (4-12) or (4-13)
3. Calculate Q_{NM}^{-1} and Q_{NA}^{-1}
4. Calculate $Q_{AM} = Q_{NM}\, Q_{NA}^{-1}$
5. Generate β_N using equation (4-16)
6. Calculate $B_M = Q_{NM} Q_{NM}^{-1}$

7. Generate $\bar{\ell}_A$ from β_o of the Cowell or Adams-Moulton method
8. Calculate $\bar{\ell}_M = Q_{AM}\,\bar{\ell}_A$

Everything is generated within the program except the value of β_o which* is input. For our calculations, we used β_o's given by Maury and Brodsky.**

C 2. COMPUTER PROGRAM SUBROUTINES

C 2.1 GEAR

PURPOSE: This is the driver program which makes use of most of the subroutines and generates the necessary coefficients. The input values are coded in the main program. For different runs to obtain different order coefficients, the input values have to be changed in the main program. More details can be seen in the program usage. As we have noted, there are two programs to generate Class I and Class II coefficients. Both programs have the same main routine with very little modification.

INPUT: k, β_o. k is the step number of the method and β_o determines the method.

---

* β_o defined by equation (4-2) for the standard corrector method.

** Maury, J. R., J. L., Brodsky, G. P., Cowell-Type Numerical Integration as Applied to Satellite Orbit Computation, Goddard Space Flight Center, Greenbelt, Maryland, Dec. 1969, x-553-69-46.

OUTPUT: The Matrices Q_{NM}, Q_{NA}, B_N, Q_{NM}^{-1}, Q_{NA}^{-1}, Q_{AM}, B_M, $\bar{\ell}_A$, $\bar{\ell}_M$

B_M and $\bar{\ell}_M$ give the necessary coefficients for the method. More details about the printed output is given in paragraph C 3.

SUBROUTINES USED: QNMMTX, QNAMTX, BNMTRX, MATINV, MPROD

DESCRIPTION: All operations done on the floating point variables are done in double precision and all the computable qualities are stored in double precision. The input to each subroutine and the output from each subroutine is passed through the argument of the call statement. This is explained in the calling sequence of each subroutine. The method involved in the computation is explained in section 4. All mathematical equations used in the program are also given in section 4.

C 2.2 QNMMTX

PURPOSE: This subroutine computes the elements of the Q_{NM} matrix. Class I program makes use of the equation (4-14) and Class II program makes use of the equation (4-15). The matrix is self contained if the value of k is known.

CALLING SEQUENCE: CALL QNMMTX (KINPUT, QNM)

KINPUT is the value of k input to the subroutine and QNM is the output matrix.

DESCRIPTION: All computations are performed in double precision. Any step number matrix can be generated with proper change in the dimension statement.

C 2.3 QNAMTX

PURPOSE: This subroutine computes the elements of the QNA matrix. Class I program makes use of the equation (4-12) and Class II program makes use of the equation (4-13). The matrix is self contained if the value of k is known.

CALLING SEQUENCE: CALL QNAMTX (KINPUT, QNA)

KINPUT is the value of k input to the subroutine, and QNA is the output matrix.

DESCRIPTION: All computations are performed in double precision. Any stepnumber matrix can be generated with proper change in the dimension statement.

C 2.4 BNMTRX

PURPOSE: This subroutine computes the elements of the B_N matrix. Both Class I and Class II programs make use of the equation (4-16). The matrix B_N is the same for both Class I and Class II methods.

CALLING SEQUENCE: CALL BNMTRX (KINPUT, BN)

KINPUT is the value of k input to the subroutine, and BN is the output matrix.

SUBROUTINE USED: BINOM

DESCRIPTION: All computations are performed in double precision. This subroutine makes use of a subroutine BINOM to compute the binomial constants in calculating BN matrix.

C 2.5 BINOM

PURPOSE: This subroutine computes binomial coefficients. The subroutine BNMTRX makes use of this subroutine to compute the matrix BN.

CALLING SEQUENCE: CALL BINOM (IJ, IM, BF)

INPUT: i and j (see equation (4-16).

OUTPUT: $BF = \binom{j}{i}$

DESCRIPTION: All computations are performed in double precision.

$$\binom{j}{i} = \frac{j(j-1)\ldots(j-i+1)}{1.2.3\ldots\ i} \quad ; \ i = 1, 2, \ldots$$

and $$\binom{j}{o} = 1$$

The subroutine uses the variable IM for j and IJ for i. BF is the binomial coefficient. IJ and IM are input to the subroutine, and BF is the output.

C 2.6 MATINV

PURPOSE: This subroutine inverts a nonsingular sequare matrix. A Gaussian elimination technique is used in the inversion. This considerably eliminates error due to small values of deferminants. The routine does not check whether the matrix is singular or not.

CALLING SEQUENCE: CALL MATINV (A, N, NM, D, KSW)

A is the matrix to be inverted. A matrix is the input to the subroutine. The inverted matrix is also stored in A storage space. The output A matrix will be the inverted A matrix. N is the number of rows (or columns) in the matrix. NM is also equal to N. D is the determinant computed in the subroutine and, hence, the output from the subroutine. KSW is a dimensioned integer variable, used to store the number of rows and columns, to be used in the Gaussian elimination process.

DESCRIPTION: All computations are performed in double precision.

C 2.7 MPROD

PURPOSE: This subroutine multiplies two matrices.

CALLING SEQUENCE: CALL MPROD (N, M, N1, A, B, Z)

A and B are the two matrices to be multiplied. The product is stored in Z i.e., Z = AB. A and B are input to the subroutine. Z is the output from the subroutine. The size of the matrix A is N by M. N is the number of rows and M is the number of columns of the matrix A. The size of the

matrix B is M by N1. M is the number of rows and N1 is the number of columns of the matrix B. The output Z will have the size of N by N1.

DESCRIPTION: All computations are performed in double precision.

C 3. PROGRAM USAGE

Both the programs for Class I and Class II methods are written in double precision in FORTRAN IV to be used in the CDC 6600 computer. With very few changes the programs can be run on IBM 360 and UNIVAC 1108 computers. It is possible to obtain about 27 digits of accuracy in double precision on the CDC 6600 computer. The programs are highly modular. Subroutines can be added to or deleted from these programs easily for modifications in the computation of the coefficients.

INPUT: As noted earlier, both Class I and Class II programs have to be input in the same way. The value of k determines the step-number of the method in both Class I and Class II programs. KINPUT is the variable used in the program. It will have to be set KINPUT = k in the main program immediately after the dimension statement. The other input value is β_o which determines the method. β_o is used in the $\bar{\ell}$ (see the equation (4-6)). The value of β_o should be always positive. The sign of β_o is taken care of in the

program according to the definition. 'BETAO' is the variable used for β_o in the program. It will have to be set BETAO = β_o in the main program immediately after the statement KINPUT = k. These are the only two input information necessary for running the program in both Class I and Class II cases.

OUTPUT: The coefficients to be used in the Class I and Class II modified Gear methods are the output from the programs. The matrices used to compute the coefficients are also obtained on printed output. The matrices Q_{NM}, Q_{NA}, B_N, Q_{NM}^{-1}, Q_{NA}^{-1}, Q_{AM}, B_M and the vectors $\bar{\ell}_A$, $\bar{\ell}_M$ are the specific output from the programs. The matrix B_M and the vector $\bar{\ell}_M$ give the required coefficients for the method.

It is dimensioned to compute the coefficients up to k = 10. If needed, the dimension statements can be varied to compute the coefficients for larger-order methods. The printed output gives 16-place accuracy. Double precision on the CDC 6600 computer is about 27 digits; thus by changing the format statements for printed output more significant figures can be obtained.

APPENDIX D

SUBROUTINES DEVELOPED FOR THE GEOSTAR PROGRAM

SUBROUTINES DEVELOPED FOR THE GEOSTAR PROGRAM

D 1. INTRODUCTION

The following paragraphs describe subroutines developed for NASA's GEOSTAR program to allow use of the newly developed multistep methods. All of the new methods were implemented in the GEOSTAR subroutine CSTEP. An exception to this is the off-grid variable step program which also involves the GEOSTAR subroutine TEST. Programming was done in FORTRAN IV for the IBM 360 series computers.

At present programs for the following cases have been implemented on GEOSTAR:

1. Off-grid methods:
   a. off-grid k=6
   b. off-grid k=5
   c. off-grid k=4
   d. off-grid k=6 interpolated forces
   e. off-grid k=6 variable step
2. Modified methods k=2,---,12
3. Cyclic methods k=2,---,8.

All of the programming was done within CSTEP and TEST, thereby minimizing interface problems with the rest of GEOSTAR. Data inputs are basically the same as with the present version of GEOSTAR. For off-grid methods the inputs are exactly the same once the new version of CSTEP is inserted into the GEOSTAR program. For the modified methods the user has to input the desired value of k. For the cyclic methods the user has to input the value of k and the desired set of coefficients since there may be several methods for a given value of k. The output for all methods is identical with that presently produced by GEOSTAR.

D 2. OFF-GRID METHODS

D 2.1 Computation Procedures

Let t_{n+1} be the time point at which the satellite position and velocity is to be computed. Letting $\ddot{\overline{X}}_i$ be satellite acceleration vector at time t_i, scaled by the factor h^2, the predicted position and velocity vectors are computed

Class II

$$\overline{X}^{(0)}_{n+1} = \sum_{i=1}^{k+1} a_i \overline{X}_{n-i+1} + \sum_{i=1}^{k+1} b_i \ddot{\overline{X}}_{n-i+1} \qquad \text{(D-1)}$$

Class I

$$\dot{\overline{X}}^{(0)}_{n+1} = \sum_{i=1}^{k+1} a^*_i \dot{\overline{X}}_{n-i+1} + \frac{1}{h}\left[\sum_{i=1}^{k+1} b^*_i \ddot{\overline{X}}_{n-i+1}\right] \qquad \text{(D-2)}$$

Class II

$$\overline{X}^{(0)}_{n+1-\theta} = \sum_{i=1}^{k+1} \tilde{a}_i \overline{X}_{n-i+1} + \sum_{i=1}^{k+1} \tilde{b}_i \ddot{\overline{X}}_{n-i+1} \qquad \text{(D-3)}$$

Class I

$$\dot{\overline{X}}^{(0)}_{n+1-\theta} = \sum_{i=1}^{k+1} \tilde{a}^*_i \dot{\overline{X}}_{n-i+1} + \frac{1}{h}\left[\sum_{i=1}^{k+1} \tilde{b}^*_i \ddot{\overline{X}}_{n-i+1}\right] \qquad \text{(D-4)}$$

and corrections are given by:

Class II

$$\overline{X}^{(1)}_{n+1} = \sum_{i=1}^{k} \alpha_i \overline{X}_{n-i+1} + \sum_{i=1}^{k+1} \beta_i \ddot{\overline{X}}_{n-i+2} + \beta_\theta \ddot{\overline{X}}_{n+1-\theta} \qquad \text{(D-5)}$$

Class I

$$\ddot{\bar{X}}^{(1)}_{n+1} = \sum_{i=1}^{k} \alpha_i^* \dot{\bar{X}}_{n-i+1} + \frac{1}{h}\left[\sum_{i=1}^{k+1} \beta_i^* \ddot{\bar{X}}_{n-i+2} + \beta_\theta^* \ddot{\bar{X}}_{n+1-\theta}\right] \qquad \text{(D-6)}$$

For k=6, $\beta_1^*=0$ in the Class I corrector so it is not necessary to calculate $\ddot{\bar{X}}^{(0)}_{n+1}$. For k=4, $\beta_1 = \beta_1^*=0$, so the calculation of $\bar{X}^{(0)}_{n+1}$ and $\ddot{\bar{X}}^{(0)}_{n+1}$ are eliminated.

Because of these differences, it was felt to be more efficient to prepare a separate version of CSTEP for each case k=4, 5, and 6 rather than combining them into a single program. If later this is found to be inconvenient the methods can be incorporated into a single program without too much effort. The actual computation sequence is as follows:

<u>for k=6,</u>

Predict $\bar{X}^{(0)}_{n+1}$

Predict $\bar{X}^{(0)}_{n+1-\theta}$

Predict $\dot{\bar{X}}_{n+1-\theta}$

CALL FRCS and obtain $\ddot{\bar{X}}_{n+1-\theta}$

Correct $\dot{\bar{X}}^{(1)}_{n+1}$

CALL FRCS and obtain $\ddot{\bar{X}}_{n+1}$

Correct $\bar{X}^{(1)}_{n+1}$

CALL FRCS for final value of $\ddot{\bar{X}}_{n+1}$

<u>for k=5,</u>

Predict $\bar{X}^{(0)}_{n+1}$

Predict $\bar{X}^{(0)}_{n+1-\theta}$

Predict $\bar{\dot{X}}^{(0)}_{n+1}$

Predict $\bar{\dot{X}}^{(0)}_{n+1-\theta}$

CALL FRCS and obtain $\bar{\ddot{X}}_{n+1-\theta}$

CALL FRCS and obtain $\bar{\ddot{X}}_{n+1}$

Correct $\bar{X}^{(1)}_{n+1}$

Correct $\bar{\dot{X}}^{(1)}_{n+1}$

CALL FRCS and obtain final value of $\bar{\ddot{X}}_{n+1}$

<u>for k=4,</u>

Predict $\bar{X}^{(0)}_{n+1-\theta}$

Predict $\bar{\dot{X}}^{(0)}_{n+1-\theta}$

CALL FRCS and obtain $\bar{\ddot{X}}_{n+1-\theta}$

Correct $\bar{X}^{(1)}_{n+1}$

Correct $\bar{\dot{X}}^{(1)}_{n+1}$

CALL FRCS and obtain final value of $\bar{\ddot{X}}_{n+1}$

D 2.2 Variable Definitions

The following FORTRAN variables have been used in the program.

Variable Name	Definition
X(40,3,20), XD(40,3,20), XDD(40,3,20) Defined in Common Block /WORKER/	These are the GEOSTAR variables for the position, velocity, and acceleration vectors $\bar{X}_i$, $\dot{\bar{X}}_i$, $\ddot{\bar{X}}_i$ respectively. For these variables, the first index refers to location in the position, velocity and acceleration arrays. The second index refers to the x,y, or z coordinate. The third index indicates which variational equations are being solved, it is set equal to 1 for the equations of motion.
XTH(3)	Current off-grid position, $\bar{X}_{n+1-\theta}$. The index refers to the x,y, or z coordinate.
XDTH(3)	Current off-grid velocity, $\dot{\bar{X}}_{n+1-\theta}$. The index refers to the x,y, or z coordinate.

XDDTH(3)	Current off-grid acceleration, $\ddot{\bar{X}}_{n+1-\theta}$. The index refers to the x,y, or z coordinate.
NN	Pointer to position in the position, velocity, acceleration array. NN+1 = KI(1) in GEOSTAR.
HH	Step size used in integration procedure. HH = CDEL(1) in GEOSTAR.
K	The step number of the method set either to 6,5, or 4.
K1	k+1; number of past values used.
THETA	The off-grid parameter, set equal to 0.21 for k=6, 0.15 for k=5, 0.3 for k=4.
S(6)	Temporary storage vector to save partial sums used for the predictor and corrector calculations.

SAVE(3,3) — Temporary storage matrix for $\overline{X}, \dot{\overline{X}}$, and $\ddot{\overline{X}}$, used when calling FRCS at the off-grid point. The first index indicates whether the variable is a position, velocity, or acceleration value. The second index refers to the x,y or z coordinate.

TIME — Temporary storage variable, used to save the value of T(1) when calling FRCS at the off-grid point.

FRCS — Subroutine in GEOSTAR which evaluates the force model for the equations of motion. The value of XDD( ) supplied by FRCS is $h^2\ddot{\overline{X}}_{n+1}$ or $h^2\ddot{\overline{X}}_{n+1-\theta}$ depending on the value of T(1) and XD( ) and X( ).

The variables used for the k=6, k=4, and k=5 versions of CSTEP are all the same except for values of k, and THETA. The coefficients for the method are

stored in the common block /COEFFS/ which has the following variables for k=6.

Common Block /COEFFS/

Variable Name	Definition
PRED(7,6)	Matrix of predictor coefficients. The first index refers to the i index of equations (D-1) - (D-4). The second index refers to set of coefficients being used., i.e. 1 refers to a's, 2 refers to b's, 3 refers to a*'s, 4 refers to b*'s, etc.
CORR(7,4)	Matrix of corrector coefficients. The indexes are defined the same as for the predictor coefficients.
BETATH(2)	Vector of off-grid acceleration coefficients. The first value is that for Class I, β_θ^*, the second value that for Class II, β_θ.

The same definitions hold for k=4 and k=5 except that the arrays have different dimensions. Thus for k=4, we have PRED (4,4), CORR(5,4) and for k=5 PRED(6,8) and CORR(6,4).

D 2.3 Off-Grid Method With Interpolated Forces

A version of CSTEP has been prepared in which the off-grid acceleration value is interpolated rather than calculated using FRCS. This method is explained in section 2.5. The implemented version uses the off-grid k=6 method, a 9th degree Lagrange interpolating polynomial, and an additional Hermite predictor for calculating $\dot{\bar{X}}^{(0)}_{n+1}$. The computation sequence is as follows:

For the 1st 9 Points

. Predict $\bar{X}^{(0)}_{n+1}$

. Predict $\bar{X}_{n+1-\theta}$

. Predict $\dot{\bar{X}}_{n+1-\theta}$

. CALL FRCS and obtain $\ddot{\bar{X}}_{n+1-\theta}$

. Correct $\dot{\bar{X}}^{(1)}_{n+1}$

. CALL FRCS and obtain $\ddot{\bar{X}}_{n+1}$

. Correct $\bar{X}^{(1)}_{n+1}$

. Set ISWT(12) = TRUE

. CALL FRCS and obtain the two-body final value of $\ddot{\bar{X}}_{n+1}$

. Set ISWT(12) = FALSE

. CALL FRCS and obtain the final full-forces value of $\ddot{\bar{X}}_{n+1}$

. Calculate the perturbed component of $\ddot{\bar{X}}_{n+1}$ and save it

For The Rest of The Integration Interval

. Predict $\bar{\bar{X}}^{(0)}_{n+1}$

. Predict $\bar{\bar{X}}^{(0)}_{n+1-\theta}$

. Predict $\dot{\bar{X}}_{n+1-\theta}$

. Set ISWT(12)=FALSE

. CALL FRCS and obtain the two-body value of $\ddot{\bar{X}}_{n+1-\theta}$

. Predict $\dot{\bar{X}}_{n+1}$ using an additional Hermite predictor

. CALL FRCS and obtain the two-body value of $\ddot{\bar{X}}_{n+1}$

. Set ISWT(12) = TRUE

. CALL FRCS and obtain the full-forces value of $\ddot{\bar{X}}_{n+1}$

. Calculate the perturbed component of $\ddot{\bar{X}}_{n+1}$

. Interpolate for the perturbed component of $\ddot{\bar{X}}_{n+1-\theta}$ using perturbed components from 9 previous back points and the perturbed component of $\ddot{\bar{X}}_{n+1}$

. Calculate the full-force value of $\ddot{\bar{X}}_{n+1-\theta}$ by adding the two-body component and the interpolated perturbed component

. Correct $\dot{\bar{X}}^{(1)}_{n+1}$

. CALL FRCS and obtain $\ddot{\bar{X}}_{n+1}$

. Correct $\bar{\bar{X}}^{(1)}_{n+1}$

. CALL FRCS and obtain the final full-forces value of $\ddot{\bar{X}}_{n+1}$

. Set ISWT(12) = TRUE

. CALL FRCS and obtain the two-body final value of $\ddot{\bar{X}}_{n+1}$

. Calculate the final value of the perturbed component of $\ddot{\bar{X}}_{n+1}$

The variables used in the program are the same as those used for the regular k=6 off-grid method (see paragraph D.2.2). Additional variables used are defined as follows:

Variable Name	Definition
M	Counter, initialized to zero.
BDY(3)	Vector of two-body component of $\ddot{\bar{X}}_{n+1}$ The index relates to the x, y, and z components.
THBDY(3)	Vector of 2-body component of $\ddot{\bar{X}}_{n+1-\theta}$ The index relates to the x, y, and z components.
PRTB(10,3)	Vector of perturbed component of $\ddot{\bar{X}}_{n+1}$ The first index relates to position in the array. The second index relates to the x, y, and z components.
PRTBN2(3)	Interpolated value of the perturbed component of $\ddot{\bar{X}}_{n+1-\theta}$ The index refers to the x, y, and z components.
LAGRAN(10)	Coefficients of the Lagrange interpolating polynomial.

Variable Name	Definition
A(7), B(7)	Coefficients for the prediction of $\dot{\bar{X}}^{(0)}_{n+1}$, "a" coefficients are given by A(7), and the "b" coefficients are given by B(7). The predictor equation is given by:

$$\dot{\bar{X}}^{(0)}_{n+1} = \sum_{i=0}^{6} a_i \dot{\bar{X}}_{n-i} + \frac{1}{h}\left[\sum_{i=0}^{6} b_i \ddot{\bar{X}}_{n-i}\right]$$

D.2.4 Off-Grid Method With Variable-Step Procedure

The variable-step procedure discussed in paragraph 2.6 has been implemented in GEOSTAR. The implementation involves the subroutines CSTEP and TEST. In CSTEP the local error calculation is performed. In TEST, the magnitude of the error is tested and when necessary the step size is changed. The method is programmed correctly, however, the values of TOL1, TOL2, and TOL3 have to be adjusted to off-grid characteristics before reasonable results can be obtained with the program. The variable-step procedure has been implemented for the k=6 off-grid method. The computation sequence is the same as for the k=6 off-grid method except that at the end the local error is computed using the results of paragraph 2.6. The computation sequence for the error estimation calculation is as follows:

Correct $\bar{\bar{X}}^{(1)}_{n+1}$

CALL FRCS and obtain the final value of $\ddot{\bar{X}}_{n+1}$

Calculate $ERR_i = (X_i^{(1)} - X_i^{(0)} + R_i)\dfrac{C_{p+1}}{C^*_{p+1} - C_{p+1}}$ for i = x, y, z

Calculate $RTOT = |ERR_x| + |ERR_y| + |ERR_z|$

Calculate R_i using equation (2.6-15)

Set SPO(1) = RTOT

C_{p+1} and C^*_{p+1} are defined by equation (2.6-22) for the k=6 corrector and predictor respectively.

In TEST everything is the same as before, except that SPO(1) is set equal to SUM and the step size is changed, using the equation:

$$\text{New step size} = \text{old step size} \left(\frac{\text{TOL3}}{\text{SUM}}\right)^{\frac{1}{12}}$$

In CSTEP the variable definitions are the same as for k=6, except for the following additional variables;

Variable Name	Definition
G(7)	The coefficients for calculating the residual error term as given in (2.6-15). The values are input as data.
ERR(8,3)	The error array. The first index indicates position. The second relates to the x, y, or z component. Thus ERR(8,2) is the the error at $\bar{X}_{n+1}$ for the y component.
RR(3)	The residual error term from past back points. The index refers to the x, y, or z component.
CONST	The value of $\frac{C_{p+1}}{C^*_{p+1}-C_{p+1}}$ The value of CONST is input as data.

D 3. MODIFIED METHODS

D 3.1 Computation Procedure

As was the case for off-grid methods, t_{n+1} is the time point at which the satellite position and velocity are to be computed. The acceleration at time t_i scaled by h^2 is $\ddot{\bar{X}}_i$. The predicted position and velocity vectors are computed from:

Class II

$$\bar{X}^{(o)}_{n+1} = \sum_{i=1}^{k} a_i \bar{X}_{n-i+1} + \sum_{i=1}^{k} b_i \ddot{\bar{X}}_{n-i+1} \qquad \text{(D-7)}$$

Class I

$$\dot{\bar{X}}^{(0)}_{n+1} = \sum_{i=1}^{k} a^*_i \dot{\bar{X}}_{n-i+1} + \frac{1}{h}\left[\sum_{i=1}^{k} b^*_i \ddot{\bar{X}}_{n-i+1}\right] \qquad \text{(D-8)}$$

The points $\bar{X}^{(0)}_{n+1}$, $\dot{\bar{X}}^{(0)}_{n+1}$ along with the previous k-1 points are corrected as follows

Class II

$$\bar{X}_{n-i+2} = \bar{X}_{n-i+2} + c_i F \quad i = 1,2, \ldots k \qquad \text{(D-9)}$$

Class I

$$\dot{\bar{X}}_{n-i+2} = \dot{\bar{X}}_{n-i+2} + c^*_i F^* \quad i = 1,2, \ldots k \qquad \text{(D-10)}$$

where

$$F = h^2\ddot{\bar{X}}_{n+1}(0) - \ddot{\bar{X}}_{n+1} \tag{D-11}$$

$$F^* = h\ddot{\bar{X}}^*_{n+1}(0) - \frac{1}{h}(\ddot{\bar{X}}_{n+1}) \tag{D-12}$$

and

$$h^2\ddot{\bar{X}}_{n+1}(0) = \sum_{i=1}^{k} \gamma_i \bar{X}_{n-i+1} + \sum_{i=1}^{k} \delta_i \ddot{\bar{X}}_{n-i+1} \tag{D-13}$$

$$h\ddot{\bar{X}}_{n+1}(0) = \sum_{i=1}^{k} \gamma_i^* \dot{\bar{X}}_{n-i+1} + \frac{1}{h}\left[\sum_{i+1}^{k} \delta_i^* \ddot{\bar{X}}_{n-i+1}\right] \tag{D-14}$$

These equations have been implemented in CSTEP for k=4 and 6*. The following computation sequence is used in the program.

Predict $\bar{X}_{n+1}(0)$

Predict $\dot{\bar{X}}_{n+1}(0)$

Calculate $h^2\ddot{\bar{X}}_{n+1}(0)$

Calculate $h\ddot{\bar{X}}_{n+1}$

*The program is developed for k=2 through 12. However, at present coefficients have only been used for k=4 and 6.

CALL FRCS and obtain $\ddot{\bar{X}}_{n+1}$

Calculate F and F*

Correct $\bar{X}_{n+1}, \ldots, \bar{X}_{n-k+2}$

Correct $\dot{\bar{X}}_{n+1}, \ldots, \dot{\bar{X}}_{n-k+2}$

CALL FRCS for final value of $\ddot{\bar{X}}_{n+1}$

D 3.2 Variable Definitions

The following FORTRAN variables have been used in the program.

Variable Name	Definition
X(40,3,20), XD(40,3,20) XDD(40,3,20) Defined in COMMON Block /WORKER/	These are the GEOSTAR variables for position, velocity, and acceleration. They are discussed in more detail in D 2.2.
H2XDD(3)	The vector $h^2\ddot{\bar{X}}_{n+1}(0)$. The index refers to the x,y, and z components.
HXDD(3)	The vector $h\ddot{\bar{X}}_{n+1}(0)$. The index refers to the x,y, and z components.

NN	Pointer to position in the position, velocity, and acceleration arrays NN+1 = KI(1) in GEOSTAR.
HH	Step size used in the integration procedure. HH = CDEL(1) in GEOSTAR.
K	The step number of the method.
S(8)	Vector used to hold partial sums while calculating $\overline{X}$ and $\dot{\overline{X}}$.
F	Corresponds to F defined by equation (D-11).
FSTAR	Corresponds to F* defined by equation (D-12).

The coefficients of the methods are stored in the common block/COEFFS/. This consists of the variables ALPHA(77), BETA(77), ALPHAS(77), BETAS(77), GAMMA(77), DELTA(77), GAMMAS(77), DELTAS(77), EL(77), ELS(77). These are

a, b, a*, b*, γ, δ, γ*, δ*, c, c* respectively (i.e., the coefficients defined by equations (D-7) - (D-10)). The pointer L is used to obtain the required coefficients. It is calculated using the relation L = (k)(k-1)/2. The program is written for k=2, ..., 12. However, at this time only coefficients for k=4 and 6 are used.

D 3.3 Data Input

Besides the normal control cards for GEOSTAR the value of K must be specified. It is read in as data using data set reference number 4. The value of K is placed in the first two columns with format I2. At present there are coefficients for K=4 and 6.

D 4 CYCLIC METHODS

D 4.1 Computation Procedures

Let t_{m+1} be the time point at which the satellite position and velocity are to be computed. Letting $\ddot{\bar{X}}_i$ be the satellite acceleration vector at time t_i, scaled by the factor h^2, the predicted position and velocity vectors are computed from:

Class II

$$\bar{X}^{(0)}_{n+m} = \sum_{i=1}^{k+1} a_i^{(m)} \bar{X}_{n+m-i+1} + \sum_{i=1}^{k+1} b_i^{(m)} \ddot{\bar{X}}_{n+m-i+1} \qquad \text{(D-15)}$$

Class I

$$\dot{\bar{X}}^{(0)}_{n+m} = \sum_{i=1}^{k+1} a_i^{*(m)} \dot{\bar{X}}_{n+m-i+1} + \frac{1}{h}\left[\sum_{i=1}^{k+1} b_i^{*(m)} \ddot{\bar{X}}_{n+m-i+1}\right] \qquad \text{(D-16)}$$

The mth corrector is computed from:

Class II

$$\bar{\bar{X}}^{(1)}_{m+m} = \sum_{i=1}^{k} a_i^{(m)} \bar{\bar{X}}_{m+m-i+1} + \sum_{i=1}^{k+1} B_i^{(m)} \ddot{\bar{X}}_{m+m-i+2} \qquad \text{(D-17)}$$

Class I

$$\dot{\bar{X}}^{(1)}_{n+m} = \sum_{i=1}^{k} \alpha_i^{*(m)} \dot{\bar{X}}_{n+m-i+1} + \frac{1}{h} \left[\sum_{i=1}^{k+1} \beta_i^{*(m)} \ddot{\bar{X}}_{n+m-i+2} \right] \qquad \text{(D-18)}$$

where m=1→k and then is recycled.

The presently developed k=4 methods use the same predictor at every step. However the program is developed to use m different predictors if this is desirable. Hence, the notation of equations (D-15) and (D-16).

The computational sequence is given by:

Input coefficients of desired method (e.g., $\alpha_i^{(m)}, \beta_i^{(m)}$ i=1,...,k m=1,...,k)

Predict $\bar{\bar{X}}^{(0)}_{n+m}$

Predict $\dot{\bar{X}}^{(0)}_{n+m}$

CALL FRCS and obtain $\ddot{\bar{X}}_{n+m}$

Correct $\bar{\bar{X}}^{(1)}_{n+m}$

Correct $\dot{\bar{X}}^{(1)}_{n+m}$

CALL FRCS for final value of $\ddot{\bar{X}}_{n+m}$

D 4.2 Variable Definitions

The following FORTRAN variables are used in the program:

D 4.2 Variable Definitions

The following FORTRAN variables are used in the program:

Variable Name	Definition
X(40,3,20),XD(40,3,20) XDD(40,3,20)	These are the GEOSTAR variables for position, velocity, and acceleration.
Defined in Common Block /WORKER/	
NN	Pointer to position in the position, velocity, and acceleration arrays. NN+1 = KI(1) in GEOSTAR.
HH	Step size used in the integration procedure. HH = CDEL(1) in GEOSTAR
K	The step number of the method. Possible values are 2,3,4,5,6,7,8*
K1	k+1,, the number of post values used.
M	A counter used to determine which set of coefficients are to be used for the current integration step; m cycles from 1→k, 1→k,... etc.

* At present only coefficients for k=4 have been used.

Variable Name	Definition
LB	Pointer to beginning of current set of coefficients being used for current integration step. $LB=\sum_{j=3}^{k-1} j*(j+1) + (M-1)*k+1$

The coefficients of the methods are stored in the common block /COEFFS/. This consists of the variables A(232), B(232), AS(232), BS(232), ALPHA(232), BETA(232), ALPHAS(232), BETAS(232). These are the coefficients $a_i^{(m)}$, $b_i^{(m)}$, $a_i^{*(m)}$, $b_i^{*(m)}$, $\alpha_i^{(m)}$, $\beta_i^{(m)}$, $\alpha_i^{*(m)}$, $\beta_i^{*(m)}$ respectively. The required values are located using the formula LB+i.

D 4.3 Data Input

In addition to the normal control cards for GEOSTAR the values of K and the cyclic method coefficients must be specified. They are input using data set reference number 4. The first card contains the value of K placed in the first two columns (I2 format). This card is followed by the cyclic method coefficients. First the Class II coefficients (a's, b's, α's, β's) and then the Class I coefficients (a*'s, b*'s α*'s, β*'s).